UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

APLICACIÓN DE TÉCNICAS ESTADÍSTICAS A REGISTROS DE RED PARA EL ANÁLISIS DE TRÁFICO Y EXPLOTACIÓN DE DATOS

Daniel Perdices Burrero

Tutor: Jorge Enrique López de Vergara Méndez

JUNIO 2018

Aplicación de técnicas estadísticas a registros de red para el análisis de tráfico y explotación de datos

Autor: Daniel Perdices Burrero
Tutor: Jorge Enrique López de Vergara Méndez

Grupo de Computación y Redes de Altas Prestaciones
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio 2018

Resumen

Resumen Hoy en día, las redes definidas por software y la virtualización se abren camino en los despliegues comerciales. Cada vez más, las empresas aprovechan las posibilidades de reconfiguración y escalabilidad de la computación en la nube para su negocio. En este contexto, las tareas de gestión de red evolucionan y se adaptan a tener que considerar simultáneamente el estado de muchos dispositivos *hardware* o virtualizados. Por tanto, la monitorización de red se tiene que adaptar a recibir esta información y ver el estado de la red en conjunto. Esto hace necesario revisar las técnicas estadísticas y los diferentes algoritmos que fundamentan la monitorización de red, de forma que sea posible explotar los registros obtenidos para analizar el tráfico.

Consecuentemente, este trabajo pretende extender las técnicas y procedimientos actuales a esta situación de monitorización pasiva en varios puntos. Se propone una herramienta de análisis de tráfico de red con diversas técnicas estadísticas que, extrayendo los datos de varios puntos, es capaz de relacionarlos y extraer información de congestión o del estado de la red a través del retardo. Esta herramienta posibilita la alimentación a sistemas de detección de anomalías o la elaboración de alertas para la reconfiguración de la red, en la que no solo se proporciona la información de que hay un patrón fuera de lo común, sino además qué zona o zonas de la red monitorizada están siendo afectadas o son responsables.

Para evaluar esta herramienta, se ha creado un entorno de pruebas experimental basado en redes definidas por software y virtualización. Así, se logra tener un entorno altamente reconfigurable en el que probar diversos despliegues de manera efectiva y con un coste muy reducido, ya que solo requiere un servidor para funcionar.

Con estos datos obtenidos, se ha podido validar la herramienta y observar como esta cumple con los objetivos establecidos. Además, se ha mostrado la utilidad de la herramienta estudiando un caso real basado en datos procedentes de un despliegue comercial.

Palabras clave Monitorización de red, medidas pasivas, latencia, múltiples saltos, selección de modelos, mediana.

Abstract

Abstract Nowadays, software defined networks and virtualization break through commercial deployments. Increasingly, companies take advantage of reconfiguration and flexible scalability of the cloud for their businesses. In this context, management tasks evolve towards considering simultaneously the state of many heterogeneous network devices. Consequently, network monitoring must adapt itself to receive all this information and consider the state of the network as a whole. This makes necessary to revisit statistical techniques and different algorithms on which network monitoring is based, in order to exploit the obtained registers to analyze the traffic.

Due to this aforementioned situation, this work pretends to extend the actual techniques to a new passive monitoring perspective with multiple points of capture. For this aim, an analysis tool based on statistical methods is proposed. This tool is able to obtain information about the state of the network or congestion by looking at different points of the network. It can also supply information of the possible issue and its location to anomaly detection system, network controllers or alarm systems, so that network can be properly reconfigured and react to these events.

To evaluate this tool, a testbed based on software defined networks and virtualization is created. With this testbed, we achieved reconfigurability with a cost-effective solution that only requires one server.

With the obtained data, the tool has been validated. We also observed that all requirements were fulfilled. Furthermore, the tool was tested with a dataset, which was extracted from a real commercial deployment.

Keywords Network monitoring, passive measurements, latency, multipath, model selection, median.

AGRADECIMIENTOS

En primer lugar, incluyo el debido agradecimiento institucional al Ministerio de Educación, Cultura y Deporte por la Beca-Colaboración (convocatoria B.O.E. del 12 de agosto de 2017) con el Departamento de Tecnología Electrónica y de las Comunicaciones de la Universidad Autónoma de Madrid, que me fue concedida para el desarrollo de este proyecto.

Además, me gustaría añadir otro agradecimiento a naudit HPCN, empresa que, a través de la Cátedra UAM-naudit, no solo ha proporcionado datos necesarios para realizar este trabajo o infraestructura para ciertas pruebas, sino que también ha ayudado a dar una motivación a todo este trabajo y las tecnologías que se originan de ello.

Por otro lado, me gustaría agradecer todo lo que me han enseñado en el área de las redes y la arquitectura a todos los miembros del grupo de investigación HPCN, tanto los *senior* (Sergio L., Javier A., Paco, Ivan, Germán, etc.) como los *junior* (Rafa, Jose, Carlos, Paula, Guillermo, etc.). De manera especial y debido a la gran ayuda que me ha prestado para enfocar algunas partes del documento, le agradezco a David Muelas la ayuda que me ha dado para poder, después de mucho esfuerzo, redactar estas palabras finales.

Agradezco también mucho a Jorge E. López de Vergara, no solo el haber sido tutor de este trabajo y ayudarme a que todo esto al final saliese hacia un cauce, sino por la oportunidad de trabajar en el grupo hace ya cerca de 2 años y de poder seguir en él, orientándome y apoyándome en mi trabajo cuando era necesario.

Me gustaría agradecer también a los compañeros del grado todos los momentos vividos, que entre prácticas y prácticas, nuestros momentos hemos tenido. En especial, a Sergio, un gran amigo y el eterno compañero de prácticas, a Diego, que espero que nos deleite otra vez con un lomo a lo Wellington y que termine pronto el doble grado, y a otros muchos: Ana, Alejandro, María, Amanda, Adrián, etc.

También, le agradezco mucho a mi familia y amigos todo el apoyo y cariño durante estos años, en especial a mi madre, que ha aguantado muchas veces mi mal humor, y mi padre, que me ha preparado muchas veces la comida, me ha venido a recoger cuando no podía más y me ha cuidado siempre.

Por último pero no menos importante, me gustaría agradecerle a Ana todo el apoyo, cariño y amor que me has dado en todo momento. Gracias por estar ahí, por aguantarme y por acompañarme en este largo camino que es la vida y en el que nos queda mucho por andar.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ALGORITMOS

GLOSARIO

CDF *Cumulative Distribution Function*. Función de distribución acumulada. Función continua por la derecha que caracteriza una distribución de probabilidad y que se define $F_X(x) = P(X \leq x)$.

CPD Centro de Procesamiento de Datos. Localización en la que las empresas concentran toda su infraestructura de sistemas informáticos y que suele tener medidas especiales para la alta disponibilidad y el correcto funcionamiento de los equipos.

ECDF *Empirical Cumulative Distribution Function*. Función de distribución acumulada empírica. Estimador natural de la función de distribución.

Flujo Conjunto de paquetes que cumple una característica en común, definido en [6], es decir con la misma cuádrupla y en el mismo intervalo temporal.

GEV *Generalized Extreme Value*. Distribución generalizada de valores extremos. Modelo paramétrico empleado para modelar valores extremos o máximos y mínimos de fenómenos aleatorios.

HSIC *Hilbert-Schmidt Independence Criterion*. Criterio de Hilbert-Schmidt de independencia. Criterio de independencia basado en funciones *kernel*.

HSM *Half-Sample Mode*. Moda de mitad de la muestra. Principio heurístico para estimar la moda que dice que esta estará en el menor intervalo que contenga a la mitad de la muestra.

KDE *Kernel Density Estimator*. Estimador de densidad kernel. Estimador de la función de densidad basado en funciones kernel.

PDF *Probability Density Function*. Función de densidad de probabilidad. Función cuya integral en un conjunto, en variables aleatorias continuas, es la probabilidad de que la variable aleatoria pertenezca al conjunto.

QoS *Quality of Service*. Calidad de servicio. Conjunto de parámetros medibles que representan el rendimiento de la red.

QQPlot *Quantile-Quantile Plot*. Gráfica cuantil-cuantil. Gráfica que muestra puntos cuya coordenada X es el cuantil k de la distribución y en la Y es el estadístico de orden correspondiente a ese cuantil k; de tal manera que si los datos se ajustan a la distribución, se espera ver una relación lineal $Y = X$.

RTT *Round-Trip Time*. tiempo de ida y vuelta de un paquete, tal y como esta definido en [1].

SDN *Software Defined Networks*. Redes definidas por software.

1 Introducción

1.1 Motivación

Hoy en día, es cada vez más común tener equipamiento de red (conmutadores o *switches*, enrutadores o *routers*, balanceadores de carga) cada vez más inteligentes y dotados de capacidades de progamabilidad [20]. Estas tecnologías, avaladas por la Linux Foundation, están cada vez más presentes en los equipos de red de los fabricantes [7]. Además, están surgiendo equipos de red que permiten ejecutar software de propósito general, en ciertos casos, incluso permiten incluir virtualización dentro de ellos. Las redes definidas por software (*Software Defined Network*, SDN), que constituyen una parte troncal de la infraestructura de los sistemas *cloud*, están dotadas de funcionalidades aún más avanzadas en el plano de control.

En este contexto, el análisis de red se ha vuelto un problema que ha de evolucionar y adaptarse a estos nuevos escenarios emergentes, en los que se puede disponer de una cantidad de datos aún mayor. De manera cada vez más usual, se puede disponer de registros de monitorización de red en muchos dispositivos, en algunos casos provenientes de ejecutar simples sistemas de capturas como `tcpdump`, de equipamiento compatible con `netflow` o IPFIX, o en otros casos provenientes de disectores de alto nivel. Actualmente, el problema de análisis de estos datos en cada punto tiene un gran interés por sí mismo, pero se empieza a ver que las necesidades de la monitorización van más allá, y es necesario no solo averiguar si la red puede experimentar una peor calidad de servicio (*Quality of Service*, QoS) sino también dónde puede estar la causa, el problema o el posible cuello de botella. Por ello, surge una nueva necesidad de intentar aprovechar la información obtenida de estos múltiples puntos de captura para obtener información sobre la red.

Consecuentemente, en este trabajo se van a explotar estos registros de red, extraídos de varios puntos de captura, con diversas técnicas estadísticas, para obtener información más detallada entre varios puntos de monitorización. Una variable de estudio que resulta especialmente de interés en este contexto es el retardo (o RTT en este caso), por lo cual en muchos contextos nos limitaremos a este caso, aunque algunas partes son extrapolables a cualquier otra métrica de red.

1.2 Descripción del problema y escenarios de uso

Una de las principales motivaciones de analizar la red por saltos, es ser capaz de descomponer el RTT en componentes más simples y relativas a cada salto (Δ_i) como se observa en la figura 1.1.

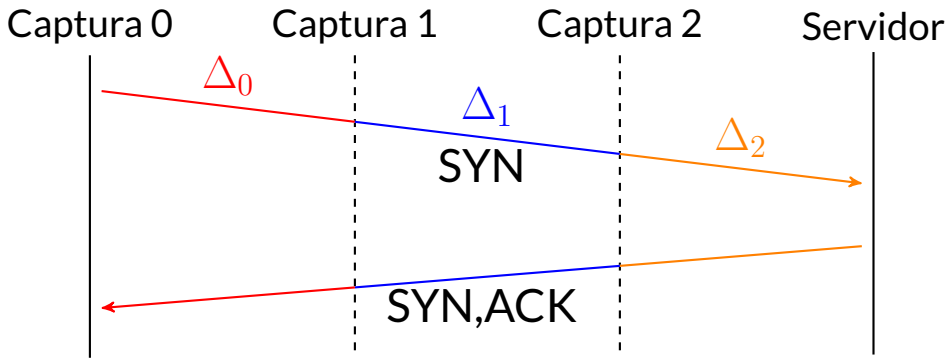

Figura 1.1: Paquetes SYN y SYN,ACK del establecimiento de conexión TCP. Se muestra el RTT total descompuesto en componentes.

Nótese que si RTT_0 es el RTT en Captura 0 (medido como la resta de los *timestamps*: ts(SYN,ACK) - ts(SYN)), entonces $\Delta_0 = RTT_0 - RTT_1$. Cada componente calculada así, a excepción de la última, contendrá la componente del RTT correspondiente al tramo de la red entre dos puntos de captura. En el caso del último salto, esa componente estará condicionada tanto por el estado de la red como por la carga del servidor, por lo que es susceptible de tener un comportamiento distinto. Esto ya permite descomponer el RTT en una componente dependiente del servidor y las restantes, en las que el tiempo de respuesta del servidor no tiene influencia alguna.

Esto nos permite dividir la red de manera lógica en distintos segmentos, en los que podemos analizar por separado el comportamiento de los equipos de red. Este nuevo enfoque que se da en este trabajo va a permitir resolver problemas de planificación o aprovisionamiento, como los que se van a detallar a continuación.

En primer lugar, el escenario de uso más común es la monitorización de un Centro de Procesamiento de Datos (CPD). Se muestra en la figura 1.2 un diagrama de una topología de ejemplo, con una mezcla de infraestructura física y virtualizada como Amazon Web Services, que es muy utilizada hoy en día para el despliegue de servicios. Como se ha motivado previamente, se va a disponer de datos de captura de distintos puntos, desde conmutadores troncales a máquinas virtuales desplegadas. Es usual en este contexto que no se puedan utilizar herramientas como `traceroute` o `ping` para evitar sobrecargar la red. Por ello, la monitorización pasiva nos va a permitir, a partir de trazas de distintos momentos del día, elaborar un diagnóstico de la red por tramos en los que vamos a poder responder diferentes preguntas como: ¿Qué tramo de la red es el más influyente? ¿Cómo impactaría al RTT un cambio o mejora en los equipos? ¿Sería dicho cambio rentable? ¿Qué partes de la red se encuentran menos utilizadas?

Otro caso de uso menos frecuente, pero incluso más importante de cara al futuro de las redes de comunicaciones, es el despliegue de funciones virtuales en

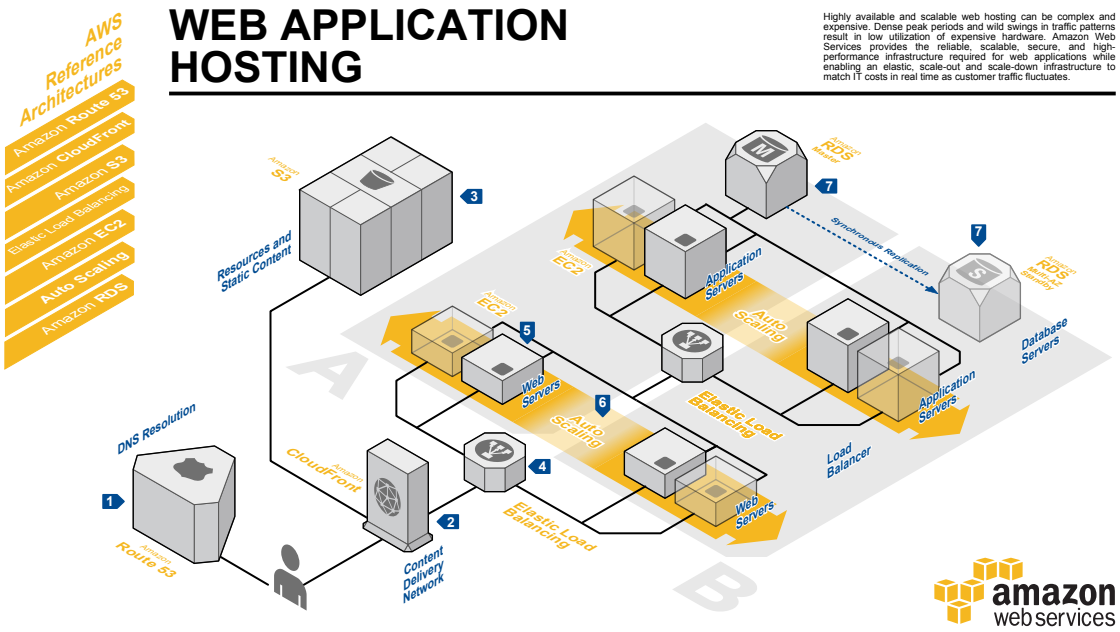

Figura 1.2: Ejemplo de despliegue de un servicio web en Amazon Web Services, extraído de https://aws.amazon.com/es/architecture/

las redes de próxima generación. Este problema tiene una relación con el anterior. En este caso, se quiere minimizar el retardo experimentado por los usuarios. Para ello hay que tener en cuenta distintos factores como carga de los nodos, distancia al nodo, etc. Mediante este método, aplicado a pequeñas muestras, podríamos ver si un nodo está muy cargado o si una serie de nodos tienen congestión simultáneamente. Este problema de despliegue de funciones virtuales es primordial para el despliegue de las futuras redes 5G.

1.3 Objetivos

El objetivo principal del trabajo es aplicar técnicas estadísticas a registros de red y explotar dichos datos, para lo que se va a desarrollar un método capaz de estimar y caracterizar la componente Δ_i que añade cada segmento de red al RTT. En primer lugar, hay que ser capaz de integrar la información de todos los puntos de captura en un solo registro, de tal manera que contenga una información completa de la conexión en todos los puntos.

Ya que la independencia de las componentes del RTT no es algo trivial y aporta desde el punto de vista teórico un gran significado, lo primero que es necesario ver es que estas componentes son independientes en los casos reales. Esto no solo es una condición muy positiva para el análisis, al poder simplificar el problema de análisis, sino que también es una condición de salud de la propia red bajo estudio, ya que lo que uno espera es que el retardo en un salto no dependa de otro. En caso contrario, esto podría indicar un fallo de configuración o un nivel alto de congestión.

Desde el punto de vista teórico, tiene importancia conocer las distribuciones de cada Δ_i, pero en concreto cobra más importancia cuando se quieren analizar factores como si existe una sufrir un retardo mayor en un salto y en el siguiente. Proporcionar un modelo permite extraer información acerca de qué se puede

considerar frecuente y qué se puede considerar poco frecuente e incluso predecir ciertos fenómenos como la probabilidad de sufrir un retardo mayor que el doble de la media o el valor más frecuente.

En conclusión, para alcanzar el objetivo principal se plantean las siguientes metas:

1. Integrar la información de los diferentes puntos de captura.
2. Extraer estimaciones de los Δ_i de forma pasiva con la información integrada.
3. Analizar las posibles relaciones de dependencia entre las posibles componentes.
4. Elaborar un algoritmo de selección de modelos evaluando distintos factores como la complejidad o la bondad de ajuste.

De esta manera, será posible explotar los registros de la red para realizar un análisis de su tráfico.

1.4 Requisitos

Se plantean a continuación los requisitos del desarrollo. Se detallan dos grupos diferenciados de requisitos, unos orientados a la propia herramienta de análisis y otros dedicados a un entorno de pruebas para probar el desarrollo. En general, se hablará de métricas de red en los requisitos, pero normalmente el análisis y los resultados, salvo que se diga lo contrario, serán sobre el RTT.

1.4.1 Análisis de tráfico

- Ser capaz de correlar la información de cada punto de captura para obtener una estructura que contenga distintos datos de una misma conexión en cada punto de captura.
- Obtener información de las distribuciones muestreadas multivariantes que resulten de integrar tráfico que provenga de un conjunto determinado de saltos.
- Ser capaz de contrastar hipótesis más propias del contexto multivariante como la independencia.
- Ser capaz de hallar la moda de una métrica de una manera robusta.
- Ser capaz de caracterizar la distribución de una métrica unimodal mediante una distribución de probabilidad.
- Ser capaz de obtener métricas o estadísticos que nos permitan evaluar el ajuste de los modelos a los datos y así poder compararlos.

1.4.2 Entorno de simulación y pruebas

- Crear un *testbed* mediante software que permita modificar métricas de red de una manera controlada.
- Ser capaz de simular topologías de red arbitrarias.

- Ser capaz de simular limitaciones de QoS mediante utilidades como `netem` o `tc`.
- Ser capaz de simular tráfico dentro de esa red, desde pruebas muy sintéticas como `ping` o `iperf`, hasta incluso introducir sistemas más complejos que intentan realizar operaciones más propias de usuarios de la red.

De estos dos grupos de requisitos, surgen dos herramientas que se van a desarrollar en este trabajo:

1. Un programa o conjunto de herramientas de análisis de datos que provean estos resultados. En este caso la visualización no es objeto de estudio de este trabajo y simplemente se busca su impresión a archivos CSV para después cargarlos en sistemas de visualización o una integración con *frameworks* de análisis de red.
2. Un programa o conjunto de herramientas de generación de tráfico con varios puntos de captura. Esta herramienta debe recibir un archivo que describa la topología de la red (nodos, tipos de nodos, enlaces entre nodos, retardos de los enlaces, ancho de banda de los enlaces). En este caso, es importante que la salida sean capturas de tráfico en las distintas interfaces de red de cada nodo, para intentar trabajar lo más cercano al caso real.

1.5 Plan de trabajo

El plan de trabajo seguido en este trabajo de fin de grado se detalla en el diagrama de Gantt de la figura 1.3, en el que en color rojo se encuentras las tareas relacionadas con la herramienta de análisis de datos y en color azul las tareas relacionadas con la creación del entorno controlado. Todas las tareas se detallan a continuación:

1. **Investigación del estado del arte**: en esta tarea se busca obtener información variada sobre diferentes técnicas de análisis de datos en el contexto del análisis de tráfico de red.
2. **Elaboración del *testbed***: el objetivo es elaborar un entorno de pruebas controlado en el que se puedan simular diferentes topologías siendo capaz de controlar parámetros de QoS de los enlaces.
3. **Análisis de datos**: Una vez se obtengan datos del primer *testbed*, se podrán analizar e intentar aplicar diferentes técnicas que sean de utilidad. El producto final de esta etapa será una primera implementación de los primeros algoritmos estadísticos junto a visualizaciones de los datos.
4. **Corrección de errores**: Se reserva una cantidad de tiempo para corregir posibles errores que no se detecten hasta que se empieza a realizar el análisis de datos. El tiempo restante se incorpora a la ampliación de la herramienta implementando más topologías.
5. **Otros casos de topología**: en esta última etapa de desarrollo del entorno de pruebas, se pretende dar soporte a ciertos elementos dentro de la topología como las bifurcaciones (a través de balanceadores de carga) y la congestión, es decir, el efecto de tráfico adicional al que nos interesa medir.

6. **Elaboración de herramientas**: en la última etapa de análisis de datos, se crearán herramientas software capaces de computar todos los algoritmos descritos y se aplicarán a diferentes conjuntos de datos obtenidos a lo largo del desarrollo así como a conjuntos de datos reales.

7. **Documentación**: Se reserva una última fase para la redacción de este documento, recopilando toda la información obtenida durante la elaboración del trabajo.

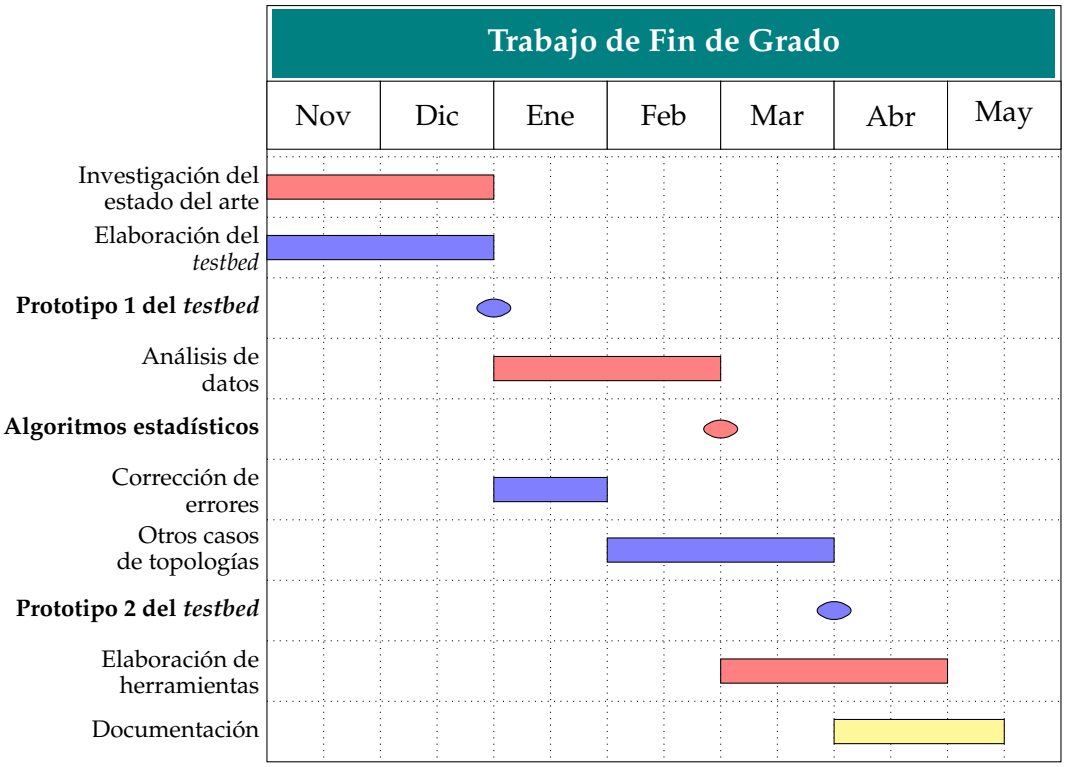

Figura 1.3: Diagrama de Gantt del proyecto

1.6 Estructura del documento

Este documento está organizado de la siguiente manera: en este capítulo 1 se da una introducción en la que se definen las metas y objetivos del trabajo. En segundo lugar, el capítulo 2 contiene un resumen del estado del arte que está relacionado con este trabajo y las principales técnicas o conceptos matemáticos en los que se fundamenta este trabajo. A continuación, el capítulo 3 contiene información de la implementación del entorno experimental y de los programas de análisis. En el capítulo 4, se muestran y analizan los resultados obtenidos en el entorno experimental y en un conjunto de datos de un CPD real. Para terminar, se presentarán las conclusiones y el trabajo futuro en el capítulo 5.

ESTADO DEL ARTE Y TÉCNICAS 2

2.1 Introducción

En este capítulo del documento se abordarán dos temas: el estado del arte y las técnicas estadísticas que se consideran de interés y serán aplicadas a este trabajo.

Por un lado, se pretende resumir el estado del arte tanto en el ámbito de las medidas de retardo de red como en el ámbito del análisis de datos de red. Ambos casos resultan ser áreas de un alto interés en las redes de comunicaciones y, por tanto, simplemente se dará una muestra de las principales referencias utilizadas como inspiración para este trabajo. Además, se mostrarán otros trabajos que resultan de gran interés para el análisis de red, tienen relación con el trabajo realizado y son fuente de inspiración para tanto este trabajo como el posible trabajo futuro que se origine a partir de este punto.

Por otro lado, se introducirán los fundamentos de ciertas técnicas estadísticas en tres partes: tests de independencia, distribuciones paramétricas y estimación de la moda. En esta sección se comenta cómo se calculan y qué significan, sin llegar a dar una explicación exhaustiva. Para pruebas o justificaciones de los resultados, se remitirá siempre a referencias de la bibliografía.

2.2 Estado del arte

2.2.1 Medidas de retardo en red

Para el caso del RTT, existen numerosos estudios y métricas diferentes. Destacan las metodologías usadas por el grupo de trabajo IPPM del IETF [15, 1], estándares que definen el concepto de retardo y especifican cómo se debe medir.

Para el caso de análisis de una red con varios nodos, en los que se quiere analizar no solo el retardo en un punto sino también determinar el retardo en cada punto de la red, existe la conocida herramienta `traceroute`, basada en monitorización activa. Sin embargo, esta herramienta sufre ciertas limitaciones ante topologías de red complejas, por lo que existen otras alternativas que extienden dicha herramienta. Por ejemplo: Paris-traceroute [2], que utiliza un

mecanismo distinto para poder ver los balanceadores de carga a nivel IP, o Pamplona-traceroute [9], que identifica las diferentes direcciones IP que están asociadas a un mismo *router*.

Para el propósito de este trabajo, no nos interesan solo las medidas activas de red, sino también las pasivas. Existen ya trabajos previos que, como nosotros, tienen como objetivo la adaptabilidad de la red, pero resultan sistemas con modelos más complejos, de los que es difícil extraer información [22]. No obstante, una parte crítica en ambos trabajos es el estimar el RTT de manera pasiva. Para ello, una manera muy simple y muy utilizada es simplemente medir la diferencia entre los *timestamps* de SYN,ACK y SYN en el establecimiento de las conexiones TCP [12]. Sin embargo, existen otras maneras de estimar el RTT [32] que, aunque se limiten a escenarios concretos, pueden ser de utilidad para extender el trabajo realizado. Además, protocolos de monitorización como NetFlow permiten estimar el RTT a través de las marcas temporales de inicio de flujo en cada sentido, lo que permite fácilmente calcular el RTT si se dispone de ambos registros de flujo de la conexión.

2.2.2 Análisis de datos de red

El análisis de los datos de red ha sido un problema muy estudiado en el pasado desde distintos puntos de vista. Uno de los principales modelos utilizados en el pasado ha sido el modelo de proceso de Poisson [8]. Este modelo, empleado en el contexto de la teoría de colas, surge de manera natural al tener colas de procesamiento muchos de los dispositivos de red. No obstante, esta teoría se basa en fuertes hipótesis sobre la distribución del tráfico, lo que en general se ha visto que no se cumple [27]. Por ello se han planteado otros modelos, que han servido de inspiración para el análisis del retardo extremo a extremo [17] o de otras variables [21].

Existen diferentes maneras de ver los datos, según el contexto a veces unas pueden ser más útiles que otras. Estudiando los datos como una serie temporal, encontramos técnicas útiles muy orientadas a la predicción y detección de anomalías. Algunos ejemplos son la utilización de modelos autorregresivos [5] o la elaboración de líneas base [35]. En este contexto, una de las tareas más comunes suele ser la búsqueda de tendencias o patrones, ya que resulta de gran utilidad para la detección de anomalías tanto *offline* como en tiempo real.

Otras de las posibles técnicas de analizar datos que ahora está tomando cada vez más fuerza es el *Functional Data Analysis* (FDA), que es capaz de estudiar los datos muestrales de ciertas métricas como funciones continuas. En este contexto, surgen ciertas técnicas que se han utilizado aquí como el *Kernel Smoothing*. Se pueden encontrar amplios trabajos previos, como [23, 24], que realizan un estudio de las diferentes técnicas de FDA y sus posibilidades en el análisis y gestión de redes, aunque los modelos son más complejos y difíciles de interpretar.

2.2.3 Metodologías de simulación de red mediante SDN

Las SDN y la virtualización a nivel de red ha sido una de los avances que más han cambiado las redes hoy en día [33]. Por ello, existe un amplio estado del arte y diferentes alternativas para probar despliegues de redes. Un ejemplo de estos sistemas es SELENA [29]. Estos sistemas suelen estar basados en

diferentes tecnologías: virtualización (e.g. VirtualBox), contenedores Linux (*Linux containers*, p.e. Docker) o espacios de nombres (*namespaces* o *cgroups*, p.e. mininet).

Del conjunto anterior, en este trabajo vamos a centrarnos en **mininet**, al ser uno de los entornos más ampliamente utilizados. Este sistema aprovecha los espacios de nombres para limitar o aislar los recursos, siendo cada *Host* un espacio de nombres distinto con sus propios procesos. Nótese que estos procesos no están aislados unos de otros, pues solo se están limitando los recursos y todos son procesos Linux corriendo en el mismo equipo.

Este entorno de emulación permite reconfiguración y mucha flexibilidad. Para el propósito concreto de crear un entorno de pruebas virtual, ya se han realizado estudios parecidos [16] en el pasado. En concreto, los autores en [25] presentan metodologías para la elaboración de estos entornos que aplican al estudio de una herramienta, aunque no en el ámbito de múltiples puntos de captura.

2.3 Técnicas estadísticas

En esta sección se dará una breve introducción a todos las técnicas estadísticas y conceptos utilizados en este documento.

2.3.1 Tests de independencia

2.3.1.1 Test χ^2 de Pearson

Este test, de principios del siglo XX, se basa en la construcción de un estadístico χ^2 a partir de la hipótesis nula de independencia [28]. Para ello, en primer lugar se construye una tabla de contingencia. Se denomina $O_{ij} = \#\{\text{Elem. en el } bin\ i \text{ de } X \text{ y } j \text{ de } Y\}$, $O_{i\cdot} = \sum_{j\in J} O_{ij}$ y $O_{\cdot j} = \sum_{i\in I} O_{ij}$ Entonces bajo $H_0 \equiv$"X, Y independientes", se tiene que el número esperado de elementos en el *bin* i de X y j de Y es $e_{ij} = N\frac{O_{i\cdot}}{N}\frac{O_{\cdot j}}{N} = \frac{O_{i\cdot}O_{\cdot j}}{N}$

Con estos cálculos, se calcula el estadístico $\chi^2 = \sum_{i\in I}\sum_{j\in J}\frac{(O_{ij}-e_{ij})^2}{e_{ij}}$, que bajo H_0 cumple que $\chi^2 \sim \chi^2_{(n-1)(m-1)}$, con (n, m) el tamaño de la tabla de contingencia.

Este test tan sencillo pero a la vez tan potente permite probar la independencia de manera bastante fácil, pero tiene inconvenientes. Por ejemplo, la necesidad de realizar una tabla de contingencia aumenta el coste computacional, pero además hace que el resultado dependa fuertemente de la construcción de la tabla, ya que si son independientes no depende de la elección de la tabla, pero en caso de que sean dependientes, la elección de la tabla puede condicionar el resultado.

2.3.1.2 Criterio de independencia de Hilbert-Schmidt (HSIC)

Para probar la hipótesis de independencia de las variables dos a dos, se busca un test distinto al ampliamente utilizado χ^2, que depende altamente de la elaboración de una tabla de contingencia. Para ello, se utiliza el test HSIC [11].

Este test se basa en espacios de Hilbert con kernel representativo (*Representing Kernel Hilbert Space*, RKHS), espacios de Hilbert en los que podemos ver una correspondencia única entre ellos y su kernel k.

Para el cálculo empírico, denominaremos $HSIC(X,Y)$ al estadístico del contraste. Este estadístico se puede computar dada una muestra bidimensional de tamaño n en función de los kernel k y l, resultando en la siguiente expresión:

$$HSIC(X,Y) = \frac{tr(KHLH)}{(n-1)^2}$$

Con $K_{ij} := k(x_i, x_j)$, $L_{ij} := l(y_i, y_j)$ y $H_{ij} := \delta_{i,j} - m^{-1}$, donde k y l son las funciones kernel. En las implementaciones populares de este test, se suele utilizar el kernel Gaussiano $k(x,y) = l(x,y) = \exp((x-y)^2/\sigma^2)$

2.3.2 Modelado mediante distribuciones de probabilidad

Una de las partes más críticas para entender el comportamiento de ciertas variables que se estudian en el análisis de red es el manejar conceptos como colas pesadas, asimetrías, etc. Por ello, modelos paramétricos muy comunes en la literatura como la distribución exponencial o los procesos de Poisson no suelen resultar adecuados en todos los contextos. A continuación, se mostrarán diferentes distribuciones de probabilidad que se han visto durante los experimentos que son de relevancia.

2.3.2.1 Distribución normal y lognormal

La distribución normal es un modelo biparamétrico que se utiliza bastante en la literatura por sus propiedades tan útiles. Tiene resultados a su alrededor, como el teorema central del límite (TCL) o como el análisis de la varianza, que la hacen una de las distribuciones más estudiadas de la estadística.

La función de densidad (*Probability Density Function*, PDF) de una variable normal $X \sim \mathcal{N}(\mu, \sigma)$ es:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

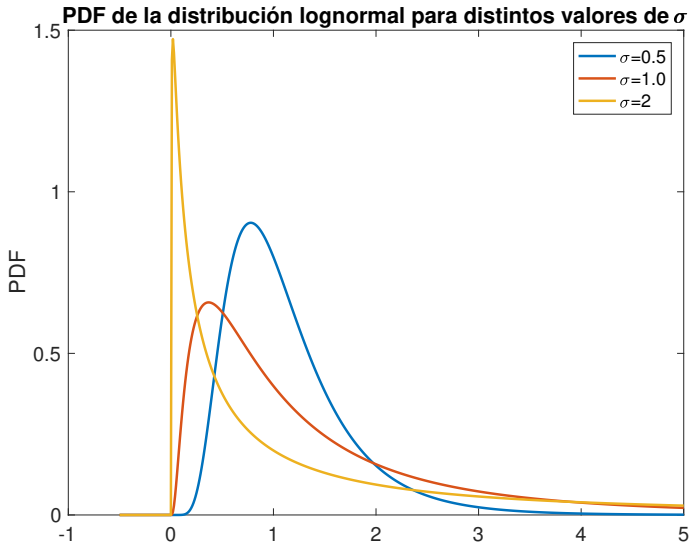

Figura 2.1: PDF de una variable lognormal para diferentes parámetros

Por otro lado, la distribución lognormal se define como la exponencial de una variable normal, esto es, $Y \sim logN(\mu, \sigma)$ si $Y = \exp(X)$ con $X \sim \mathcal{N}(\mu, \sigma)$.

Se puede comprobar que la PDF tiene la siguiente expresión:

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(log(x)-\mu)^2}{\sigma^2}\right)$$

La distribución lognormal, cuya PDF se observa en la figura 2.1, se utiliza bastante para caracterizar el producto de variables aleatorias positivas debido al TCL y a las propiedades del logaritmo respecto a la suma. Es una distribución unimodal, positiva, asimétrica y sin cola pesada. Además tiene una expresión simple y práctica para la moda: $\text{Moda}(X) = \exp(\mu - \sigma^2)$.

2.3.2.2 Distribución generalizada de valores extremos (GEV)

Esta familia de distribuciones de probabilidad se desarrolla alrededor de la teoría de grandes desviaciones y de valores extremos. En ella, se busca hallar la distribución del máximo de variables aleatorias o medir la probabilidad de que ciertos sucesos se desvíen lo suficiente de la media. Se utiliza bastante en análisis de riesgos, finanzas o hidrología. El teorema de Fisher-Tippett-Gnedenko describe el comportamiento asintótico de máximos de variables aleatorias en función de esta distribución.

Se puede comprobar que si $X \sim GEV(\mu, \sigma, \xi)$, su PDF tiene la siguiente expresión [19]:

$$f(x) = \begin{cases} \exp(-(1-\xi s)^{-1/\xi}) & si \quad \xi \neq 0 \\ \\ \exp(\exp(-s)) & si \quad \xi = 0 \end{cases}$$

Siendo $s = \frac{x-\mu}{\sigma}$ y ξ un parámetro de forma según el cual:

- Si $\xi = 0$, $X \sim$ Gumble o una distribución de valores extremos tipo I.
- Si $\xi < 0$, $Y \sim$ Fréchet con $Y = 1 + \xi s$ y parámetro $\alpha = \xi^{-}1$
- Si $\xi > 0$, $Y \sim$ Weibull opuesta $Y = -(1 + \xi s)$ y parámetro $\alpha = -\xi^{-}1$

Se muestra en la figura 2.2 la PDF para diferentes parámetros.

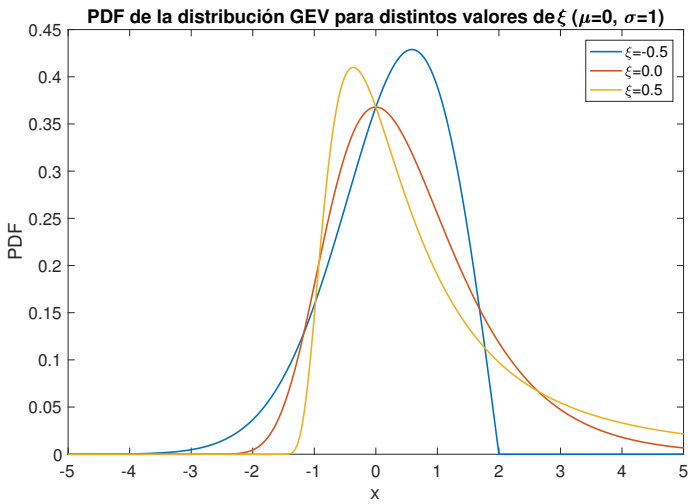

Figura 2.2: PDF de una variable GEV para diferentes valores del parámetro ξ

Esta distribución tiene la ventaja de que, dependiendo del parámetro ξ, se puede tener una cola pesada o no, lo que permite modelar variables en las que este fenómeno de la cola pesada no esté claro o sea variable.

De igual manera que antes, se tiene una expresión cerrada para la moda de la variable en función de los parámetros: $\text{Moda}(X) = \mu + \frac{\sigma}{3}[(1+\xi)^{-\xi} - 1]$.

2.3.2.3 *Distribución de Burr*

De manera similar al modelo anterior, la distribución de Burr es una familia de distribuciones de probabilidad con tres parámetros que se utiliza en el ámbito de la economía, hidrología y el análisis de riesgo.

Su formula para la PDF es:

$$f(x) = \frac{\frac{kc}{\alpha}\left(\frac{x}{\alpha}\right)^{c-1}}{\left(1+\left(\frac{x}{\alpha}\right)\right)^{k+1}} \quad x, \alpha, c, k > 0$$

En la figura 2.3 se muestran las distintas formas de esta PDF según varían k y c; α se mantiene constante por ser un parámetro de escala. Nótese que, en este caso, los parámetros no tienen una interpretación inmediata, como sí sucedía en los casos anteriores.

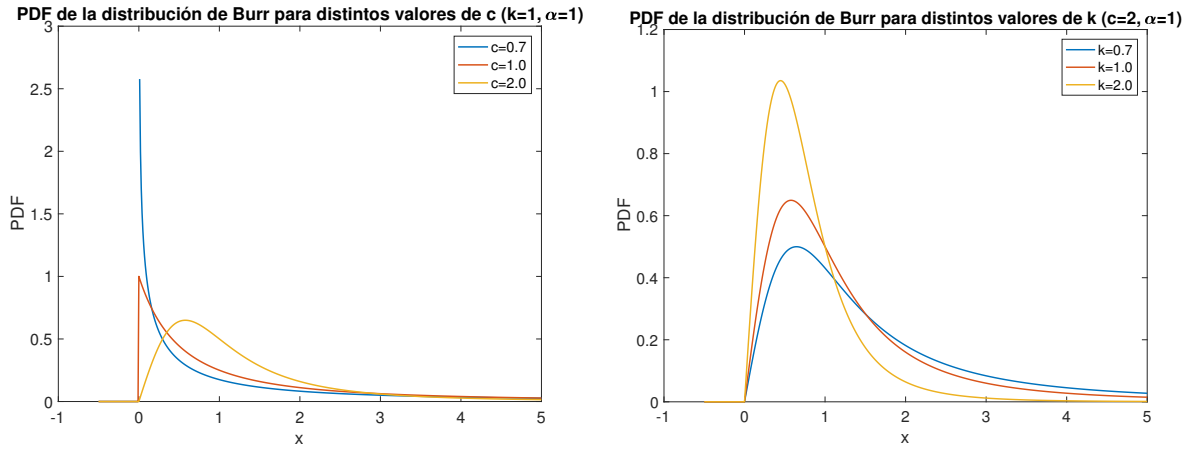

Figura 2.3: PDF de una variable Burr para distintos valores de k y c

Este modelo, pese a tener unos parámetros sin mucho significado, sigue teniendo expresiones cerradas para casi todos los parámetros de interés incluyendo la moda: $\text{Moda}(X) = \alpha\left(\frac{c-1}{kc-1}\right)^{\frac{1}{c}}$.

2.3.2.4 *Distribución α-estable*

La distribución α-estable es el modelo de probabilidad más genérico de los aquí estudiados. Esta distribución también es conocida como la familia estable o Lévy-estable de distribuciones de probabilidad. Esta distribución queda definida por cumplir la propiedad de que la combinación linear de dos variables aleatorias con esta distribución sigue también esta distribución (salvo por cambios en el parámetro de escala y en el de localización), es decir, es cerrada respecto a la combinación lineal. Nótese que la distribución normal es estable, pero

existen muchas más distribuciones que cumplen esta propiedad, por ejemplo, la distribución de Cauchy o la distribución de Lévy. Esta familia pretende agrupar la clase de variables aleatorias más grande que sea cerrada respecto a combinaciones lineales.

Esta familia de distribuciones, al estar definida de manera tan abstracta, resulta tener una gran cantidad de formas a las que puede adaptarse y puede ser de gran utilidad para modelar comportamientos muy variados. No obstante, esta gran adaptabilidad tiene un alto coste: la mayoría de las expresiones de funciones como PDF, función de distribución (*Cumulative distribution function*, CDF), media, moda o mediana no son expresables de manera analítica, esto es, no podemos obtener una expresión cerrada en función de funciones más simples, sino que tenemos que computarla de manera numérica o recurrir a valores tabulados. La única manera de caracterizar esta distribución es a través de la función característica, que sí tiene una expresión analítica.

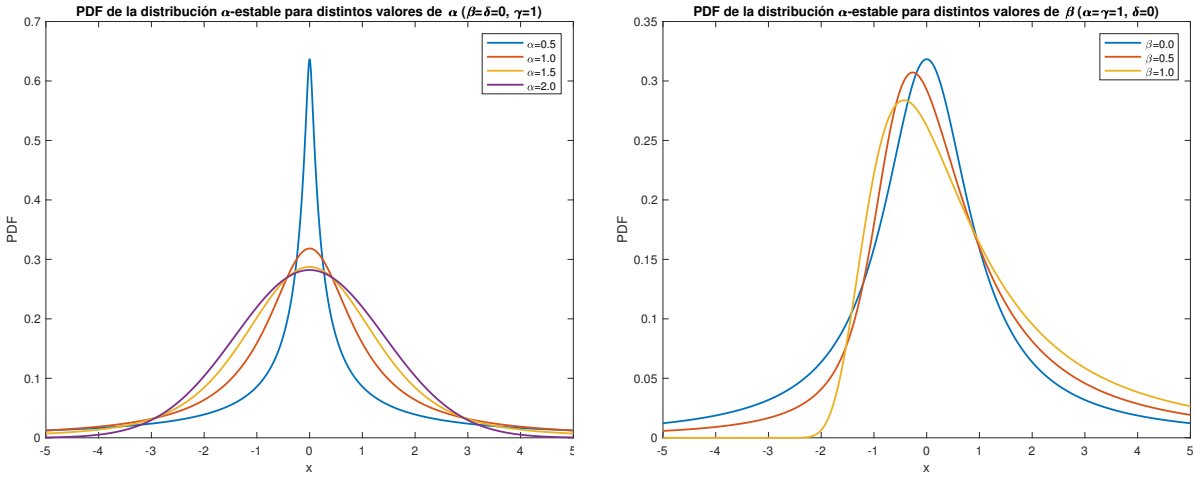

Figura 2.4: PDF de una variable α-estable para distintos valores de α y β

En la figura 2.4, se muestran diferentes PDFs de variables con distribución estable. Se puede observar la gran variedad de patrones y de comportamientos que permite modelar esta distribución.

2.3.2.5 Estimador densidad kernel de la PDF

Para estimar la PDF, hasta ahora no se ha utilizado otro método más allá del histograma. No obstante, existen estimadores no paramétricos específicos para variables aleatorias continuas. En concreto, se estudiarán los estimadores de densidad kernel (*Kernel Density Estimator*, KDE).

Este estimador consiste estimar primeramente la función característica de la variable aleatoria ($\varphi_X(t) = E[e^{itX}]$, es decir, la transformada de Fourier), que se estima por el estimador obvio ($\hat{\varphi_X}(t) = \sum_{i=1}^{N} e^{itx_i}$). A través de las fórmulas de inversión se puede obtener la PDF deseada. Esto, de manera práctica, se resume en el siguiente estimador de la PDF: $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x-x_i}{h}\right)$.

Donde K es la función kernel empleada y h es el ancho de banda del kernel. De hecho, la función kernel que minimiza el error medio cuadrático es el kernel de Epanechnikov: $K(u) = \frac{3}{4}(1-u^2)\mathbb{1}_{|u|\leq 1}(u)$

2.3.3 Estimación de la moda: algoritmo HSM

Como no siempre se va a disponer de una distribución de probabilidad que modele suficientemente bien los datos, es necesario buscar formas de extraer la moda que no realicen hipótesis sobre la distribución.

Con este propósito, existe un algoritmo recursivo presentado en [3] que computa un estadístico que converge a la moda y que está probado por ser una aproximación rápida y robusta a la estimación de la moda, un problema bastante complejo en el ámbito de las variables aleatorias continuas. Se muestra su pseudocódigo en una versión con operaciones vectoriales en el algoritmo 2.1.

Algoritmo 2.1 Algoritmo HSM

```
1:  function HSM(X, limit)
2:      if limit is 0 then
3:          return (máx(X)−mín(X))/2
4:      end if
5:      if len(X) is 1 then
6:          return X
7:      end if
8:      n ← ceil(len(X)/2)
9:      S ← X(1:end-n);
10:     EXT ← X(end-len(S)+1:end);
11:     L ← EXT - S;
12:     I = argmin(L);
13:     return HSM(X(I:I+n), limit-1);
14: end function
```

2.4 Conclusiones

En este capítulo se ha presentado un resumen de las técnicas más populares del análisis del retardo de red. Estas van desde el análisis de series temporales a su tratamiento como datos funcionales. No obstante, una aproximación que resulta interesante es el estudio de la distribución, ignorando la componente temporal o limitando el estudio a ver la distribución en un cierto intervalo temporal.

Por tanto, en la segunda sección se han presentado las diferentes componentes teóricas que fundamentan este trabajo. Destacan los modelos basados en distribuciones de probabilidad, que se van a utilizar para intentar diferenciar entre los comportamientos más y menos probables. De cada uno de estos modelos, se han presentado diferentes razones por la cual cada modelo podría servir para modelar ciertos comportamientos, se han estudiado diferentes expresiones existentes para diferentes parámetros de esas distribuciones y las ventajas de cada modelo. Además, se ha prestado mucha atención a la moda, como un estimador del valor más probable que uno va a observar y que creemos que mejor caracteriza la distribución frente a grandes desviaciones que afectan a la moda.

3 Diseño y desarrollo

3.1 Introducción

Una vez se han introducido los conceptos clave para elaborar las herramientas, es importante definir claramente el alcance del proyecto, así como la diferente funcionalidad que se va a implementar. Una vez considerados estos aspectos, queda por delante el desarrollo, en el que se han de comentar algunas decisiones que condicionan el funcionamiento de la herramienta y las razones detrás de todo este diseño.

Por tanto, en este capítulo se va a estudiar la implementación y desarrollo del entorno de red virtualizado y de la herramienta de análisis. Por la naturaleza heterogénea de ambos desarrollos, se analizarán los dos sistemas por separado como programas independientes, permitiendo esto el utilizar la herramienta de análisis por separado para despliegues reales o reutilizar el entorno experimental para probar otras herramientas.

3.2 Entorno de red virtualizado

Para la generación de tráfico de red y conjuntos de datos de prueba, se ha utilizado un entorno de red virtualizado basado en SDNs, **mininet**. Esto permite, con una única máquina, simular un entorno multicaptura que se utilizará para estudiar los flujos en distintos puntos de la red. Este tipo de metodologías de pruebas en entornos virtualizados como **mininet** permiten probar distintos despliegues y su rendimiento.

Este entorno es programable a través del API (*Application Programming Interface*) Python que tiene, y gran parte de sus nodos soportan reglas OpenFlow o reglas de `iptables`.

3.2.1 Mininet y MininetPlus

Sobre **mininet**, se ha desarrollado una nueva biblioteca, **mininetplus**, que hereda todo el API de la biblioteca original, añadiendo las siguientes mejoras:

- Se da más control sobre el proceso de nombrado de las interfaces.
- Se permite el renombrado de interfaces de "TCIntf" (interfaces que funcionan bajo `tc/netem`).
- Se añade el tipo de nodo "Router Linux", que añade funcionalidad de *routing*.
- Se añade soporte para agregar reglas a la tabla de rutas en las topologías.
- Se añade el tipo de nodo "HTTP Server", que arranca en segundo plano un servidor HTTP basado en Python.
- Se añade soporte para ejecutar más de un programa en segundo plano en un nodo.
- Se añade el tipo de nodo FlexiTOP, para la generación de tráfico con patrones realistas.
- Se añade un balanceador a nivel IP (balanceador basado en Open vSwitch con reglas OpenFlow).

3.2.2 Topologías de red básicas

Se ha realizado un diseño iterativo a la hora de probar diferentes topologías y analizar los datos. Las dos primeras topologías analizadas son la topología lineal de *switches* y la topología lineal de *routers*.

Ambas se representan en la figura 3.1 y son el mínimo entorno experimental. Se observa como los puntos de captura se configuran al final de cada uno de los enlaces. Cada uno de los enlaces será configurado mediante `tc` para que tenga un retardo fijo y distinto del resto de enlaces. Para generar flujos TCP, el cliente abrirá conexiones al servidor de manera secuencial, evitando así la más mínima saturación de los equipos.

Estas situaciones, pese a ser la más básicas, son las más sintéticas, ya que lo usual es que haya tráfico interferente y que la carga de la red no sea regular. En la siguiente sección, se introducirán mejoras sobre estas situaciones iniciales.

3.2.3 Topologías de red complejas y otros elementos

Una vez configurado el primer entorno experimental virtualizado, resulta deseable intentar mejorar las pruebas que se realizan para así acercarse en la medida de lo posible al caso real. Para ello, se van a introducir dos nuevas topologías.

En primer lugar, se considera un caso parecido a los anteriores, en el que tenemos una topología lineal. Sin embargo, se introduce tráfico adicional entre nodos que no solo van del cliente al servidor, pudiendo cargar unos *switches* más que otros y permitiendo ver distribuciones más similares a las reales, ya que lo usual es que por una red haya más tráfico, y que no todo el tráfico acabe en el mismo servidor. En este caso, no vamos a analizar el tráfico añadido, solo queremos ver cómo responden los equipos y los retardos ante esta situación más variable y con más componentes aleatorias añadidas. En la figura 3.2, se puede

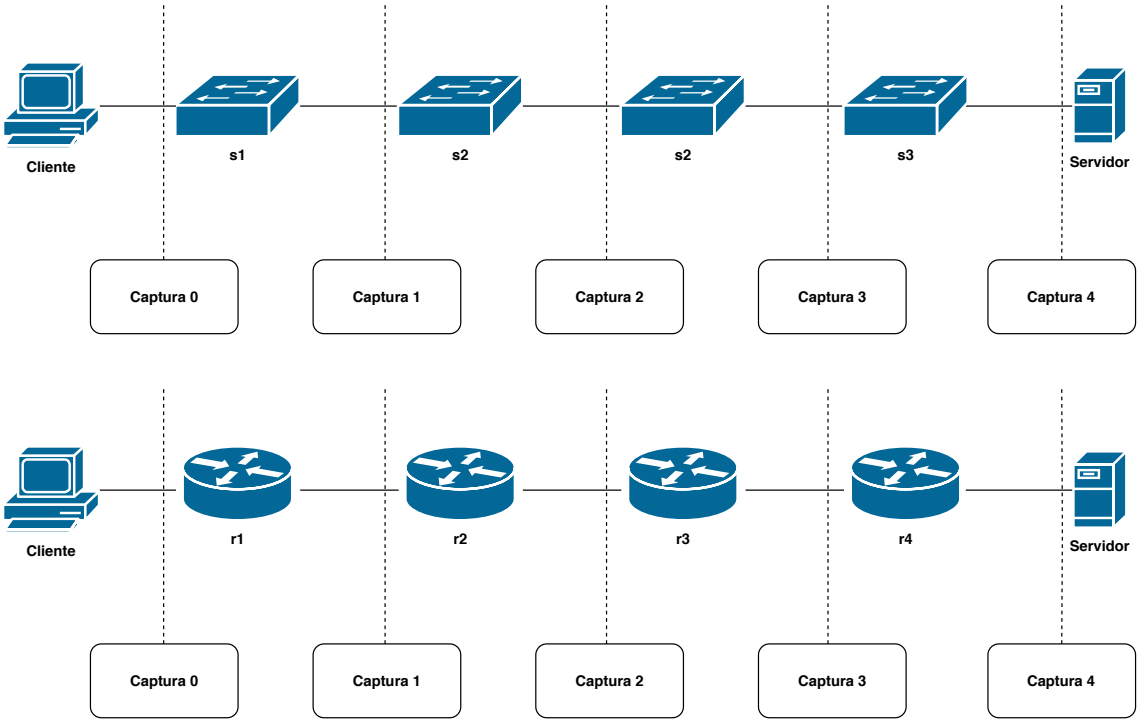

Figura 3.1: Topologías lineales con switches (arriba) y routers (abajo)

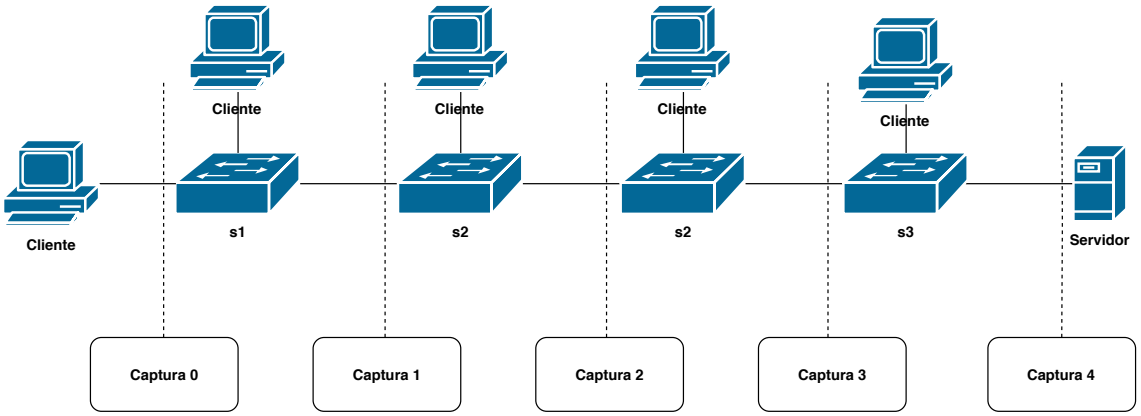

Figura 3.2: Topología lineal con congestión

observar esta topología en la que los clientes intermedios se envían mensajes entre ellos. En las pruebas realizadas, no solo se utiliza un cliente conectado a cada *switch*, sino un grupo de k clientes que envían mensajes o `ping` ICMP lanzados en intervalos aleatorios.

En general, las topologías reales no tienen por qué tener un único camino hacia un mismo nodo, de hecho, puede ser que, aunque se dirijan al mismo destino, por factores de distribución de la carga o de aseguramiento de la QoS, no vayan por la misma ruta. Por ello y para demostrar además la potencia de reconfiguración del entorno, se desarrolla una topología que tiene una bifurcación. En la figura 3.3, se observa un esquema sencillo de esta topología.

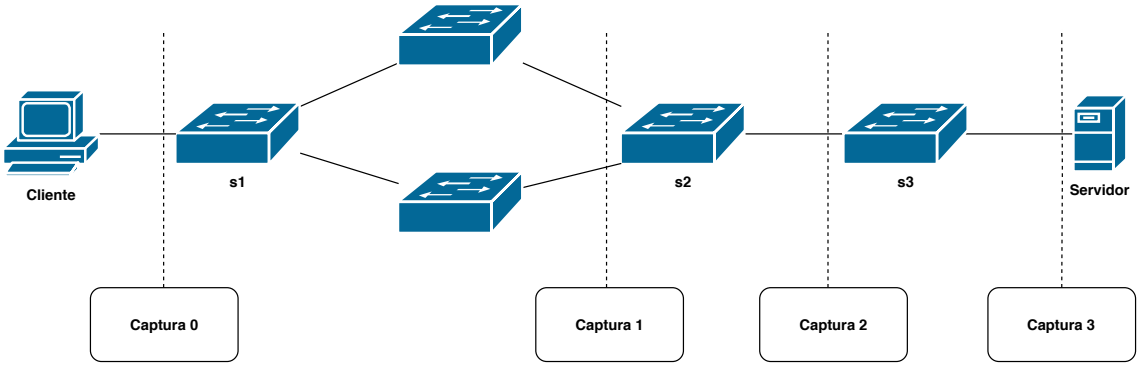

Figura 3.3: Topología lineal con una bifurcación

3.2.4 Generadores de tráfico

Una vez definidas las topologías que se quieren probar, el siguiente paso es definir la manera de generar tráfico. En este caso tenemos diferentes opciones que se presentan a continuación ordenadas de menor a mayor en complejidad:

1. **Ping ICMP**: aunque este tráfico no va a servir para ser estudiado ya que no tiene nivel TCP, permite introducir tráfico interferente y generar congestión.

2. **Cliente y servidor HTTP**: a través de simples implementaciones de un servidor HTTP y utilizando los comandos `wget` o `curl`, podemos generar conexiones HTTP de las que se capturan sus paquetes de inicio de conexión. Como optimización, ya que nos basta únicamente con abrir la conexión TCP y no es necesario terminar la transacción HTTP, utilizamos implementaciones más simples que solo abren la conexión y la cierran, mediante, por ejemplo, `netcat`.

3. **FlexiTOP y *gateway***: conectando la red virtual con el exterior a través de un NAT, podemos utilizar sistemas de medidas de QoS que van a generar patrones de tráfico más similares a los reales. En este caso, se utiliza FlexiTOP, un sistema de medidas activas de red que desarrollé para servicios Over-The-Top [30, 31]. No obstante, aunque esta parece la mejor opción, no se puede aplicar a pruebas con muchos clientes simultáneamente, ya que se abrirían muchas conexiones desde una misma IP, lo que provocaría un número muy alto de conexiones y que el proveedor de servicio bloquee la dirección IP del *host* del entorno virtualizado.

3.3 Técnicas y algoritmos de la herramienta de análisis

Esta sección englobará tanto el desarrollo e implementación de los diferentes métodos empleados en la herramienta de análisis de datos.

Por razones de portabilidad, potencia del entorno de visualización y complejidad del código, se realiza una implementación de un prototipo en Matlab para poder evaluar primeramente los resultados. Después, debido a las similitudes entre Matlab y Python (junto a las bibliotecas SciPy [13], NumPy [26] y statsmodels [34]), se implementa la versión definitiva en Python, lo que permite

integrarse con otras herramientas de análisis de tráfico y un diseño que reutilice el código ya implementado. Nótese que la herramienta definitiva está basada en bibliotecas de código abierto.

3.3.1 Herramientas y programas de procesado

3.3.1.1 Creación de superflujos

Para agrupar la información de una misma conexión vista en diferentes saltos, definiremos una estructura denominada superflujo. Esta estructura albergará diferentes flujos TCP de una misma conexión TCP, de manera que tengamos la información de cada punto de captura agrupada a nivel de conexión. Se muestra su construcción en el algoritmo 3.1:

Algoritmo 3.1 Algoritmo de construcción de superflujos

```
1:  function GETSUPERFLOWS(flows...)
2:      table ← InitializeSuperFlowsTable()
3:      for flow in flows do
4:          if flow is ip and tcp then
5:              if flow.srcPort < flow.dstPort or flow.srcPort = flow.dstPort and flow.srcIp
    < flow.dstIp then
6:                  quintuple ← (flow.srcIp, flow.srcPort, flow.dstIp, flow.dstPort,
    flow.ipProto)
7:              else
8:                  quintuple ← (flow.dstIP, flow.dstPort, flow.srcIp, flow.srcPort,
    flow.ipProto)
9:              end if
10:             table[quintuple].addFlow(flow)
11:         end if
12:     end for
13:     return table
14: end function
```

Nótese que esta aproximación muestra una simple aproximación en la que dos flujos pertenecen al mismo superflujo si y solo si comparten la misma quíntupla. Para el propósito de este trabajo, no es necesario algo más complejo, pero en ciertos despliegues de captura se almacena el tráfico de diferentes puntos de la red en el mismo entorno de captura, lo que provoca que se observen los flujos por duplicado salvo por las direcciones MAC. En proyectos más complejos, como los informes automáticos de [35], se introducen análisis más profundos que permiten reconstruir la topología de la red con algoritmos que aprovechan más información de varios de los niveles de la pila de protocolos de red (IDs de VLAN, TTL de nivel IP, etc.).

3.3.1.2 Transformación de los datos

Una de las ventajas de ver el RTT como ts(SYN,ACK) - ts(SYN), es que esto nos permite aplicar una transformación de los datos para aislar la componente del RTT visto en cada salto. Basta con definir $\Delta_j = \text{RTT}_j - \text{RTT}_{j+1}$ para todo salto j salvo el último. Para acelerar este proceso basta observar que es una transformación lineal cuya matriz es $A = (\delta_{i,j} - \delta_{i+1,j})$, lo que reduce el coste computacional.

3.3.2 Tests de independencia

Para los tests de independencia, en el caso de χ^2, no existe ningún problema más que construir la tabla de contingencia, la cual se puede hacer mirando ciertos cuantiles o valores predefinidos.

Para el test HSIC, el cálculo del estadístico no es un problema en sí, pero sí lo es encontrar el umbral de rechazo para un nivel de confianza α. Para ello, se utilizan implementaciones en Matlab [10] que siguen dos aproximaciones distintas: utilizando una propiedad asintótica a través de la distribución Gamma o mediante *Bootstrapping*. Esta implementación se puede traducir y adaptar fácilmente a código Python.

3.3.3 Ajuste de distribuciones

Para el ajuste de modelos como el normal, el lognormal o el GEV, simplemente se utilizan los métodos de ajuste disponibles tanto en Matlab como en Scipy, que están basados en estimadores de máxima verosimilitud de los parámetros.

No obstante, para el caso de la distribución α-estable, se utilizan algoritmos heurísticos que no están basados en la función de máxima verosimilitud, sino que simplemente recorren el espacio de parámetros buscando el mejor ajuste [18].

Para el cálculo de los *rankings*, surgen tres alternativas según las necesidades para evaluar qué modelo es el mejor:

1. **QQPlot y $\mathbf{R}^2$**: En este caso, simplemente calculamos los estadísticos de orden de la muestra y los juntamos con el percentil correspondiente del modelo. Estos puntos se ajustarán a una recta $Y = X$. Ajustando un modelo de regresión lineal, se espera que el $\mathbf{R}^2$ sea cercano a 1. Por ello, se elige el modelo con el $\mathbf{R}^2$ más próximo a 1.

2. **Criterio de información de Akaike (AIC)**: Este estadístico realiza una tarea similar, pero tiene en cuenta la complejidad del modelo, que se estima por el número de parámetros. Simplemente, $AIC = 2(k - \log(\hat{L}))$ con k el número de parámetros del modelo y $\hat{L}$ el máximo de la función de verosimilitud, es decir, el producto de la función de densidad del modelo ajustado en cada elemento de la muestra.

3. **Criterio de información bayesiano (BIC)**: De forma similar al anterior, este criterio introduce la complejidad de los parámetros, ponderado según el logaritmo del tamaño muestral. Esta aproximación está intrínsecamente relacionada con el AIC. Su expresión es $BIC = \log(n)k - 2\log(\hat{L})$, donde n es el tamaño muestral y el resto de variables están descritas en AIC.

Estas alternativas permiten evaluar el modelo que mejor ajusta o el modelo que mejor ajusta con la menor complejidad. No se puede decir que en todos los casos se deba elegir uno de ellos por encima del otro y dependerá del propósito concreto. Según las necesidades de monitorización y restricciones temporales, podemos mirar el AIC o el BIC para modelos *online*, ya que tienen en cuenta el coste computacional determinado por el tamaño muestral y el número de parámetros. Para modelos *offline*, podemos únicamente mirar el $\mathbf{R}^2$, ya que no tenemos restricciones tan fuertes.

3.3.4 Estimación de la moda

Estimar la moda de la muestra resulta ser un problema complejo, ya que en sí no es un concepto muchas veces bien definido en el contexto de variables aleatorias continuas. Además, puede ser tedioso la discretización de la muestra, y algo bastante discutible, ya que el resultado puede ser altamente dependiente de los *bins* elegidos para construir el histograma.

En esta sección se van a explorar algunas alternativas al proceso de estimación de la moda más allá de hallar el máximo del histograma. En todos los apartados se mencionarán las diferentes hipótesis que se asumen en cada caso. E general, se limitará siempre al estudio del caso de variables unimodales y por ello siempre hablaremos de la moda como algo único.

3.3.4.1 Estimación a través de un modelo

En las secciones anteriores se ha mencionado que existían fórmulas cerradas de la moda para ciertas distribuciones. En la tabla 3.1, se muestran estas fórmulas, cuando existen, para los casos más comunes.

Tabla 3.1: Fórmulas para la moda para ciertas distribuciones en función de sus parámetros

Distribución	**Moda**
Normal(μ, σ)	μ
LogNormal(μ, σ)	$e^{\mu-\sigma^2}$
GEV(μ, σ, ξ)	$\begin{cases} \mu + \sigma\frac{(1+\xi)^{-\xi}-1}{\xi} & \text{si } \xi \neq 0, \\ \mu & \text{si } \xi = 0. \end{cases}$
Burr(α, c, k)	$\alpha\left(\frac{c-1}{kc+1}\right)^{\frac{1}{c}}$
α-estable$(\alpha, \beta, \delta, \gamma)$	No existe fórmula cerrada

Nótese que estos estimadores de la moda son altamente dependientes de como se ajustan los datos a la distribución y, por lo tanto, solo se deben considerar cuando se observe que las muestras siguen esa distribución.

3.3.4.2 Estimación a través del KDE

Esta aproximación consiste en la más intuitiva e inmediata a la hora de estimar la moda: calcular la PDF y hallar su máximo. Una de las ventajas de esta aproximación es que no se realiza hipótesis muy fuertes sobre la distribución, salvo que el estimador KDE converja a la PDF.

En conclusión, se considera para el caso unimodal estimamos la moda mediante: $\hat{\text{Moda}}(X) = \arg\max_{x\in\mathbb{R}} \hat{f}(x)$

Se observa que el KDE nos da mucha más información sobre la distribución y se podría ser capaz de estimar incluso el número de modas de la distribución. Por ejemplo, para el kernel de Epanechnikov, se puede derivando ver que $K'(u) = \frac{3}{4}(1-2u)$ para $u < 1$.

3.3.4.3 Estimación a través del algoritmo HSM

El algoritmo HSM proporciona un método robusto y rápido de aproximar la moda. No obstante, este algoritmo no resuelve de todo el problema para ciertos casos, ya que las modas estimadas cometen un error de estimación observable.

Para solucionar esta problemática, se introduce la idea de un algoritmo HSM generalizado en el algoritmo 3.2. En este nuevo algoritmo, buscamos poder controlar el ritmo de convergencia para así lograr una mayor precisión con un coste de mayor tiempo de procesamiento, por lo que ajustaremos la parte de la recursión y de la estimación de la moda. La idea teórica es la misma: la moda de una muestra de tamaño n estará en el menor intervalo que contenga a $k < n$ elementos.

Algoritmo 3.2 Algoritmo HSM generalizado

```
function GENHSM(X, limit, ModeEst, f)
    if limit is 0 then
        return ModeEst(X)
    end if
    if len(X) is 1 then
        return X
    end if
    n ← f(len(X))
    S ← X(1:end-n);
    EXT ← X(end-len(S)+1:end);
    L ← EXT - S;
    I = argmin(L);
    return HSM(X(I:I+n), limit-1, modeEst, f);
end function
```

Donde X es la muestra ordenada, ModeEst es un estimador de la moda, por ejemplo la mediana o la media, y f es una función monótona decreciente de valores naturales. Nótese que este nuevo algoritmo incluye al caso anterior ($f(n) = \frac{n}{2}$ y ModeEst(X)=Rango(X)). Esta generalización nos permite obtener un mayor control de la rapidez o precisión del algoritmo. Ejemplos de f que resultan prácticos son:

- *(Exponencial de factor k)* $f_k(n) = \text{ceil}(kn)$ con $k < 1$
- *(Aritmética de factor k)* $f_k(n) = \text{mín}(n - k, 1)$

3.3.5 Descripción de la herramienta finalmente elaborada

Se detalla a continuación las entradas y salidas de las herramientas finales elaboradas en Python:

Entradas

1. Archivos de flujos de varios puntos ordenados

o

1. Trazas de varios puntos ordenadas

Salidas

1. **Superflujo**: Estructura que almacena un flujo con sus registros de flujo vistos en distintos puntos de captura.
2. **Correlaciones e independencia entre saltos**: condición de salud de la red.
3. **Ranking de modelos**: de una serie de modelos, el que ajusta mejor en cada salto.
4. **Moda**: valor más frecuente del Δ en cada salto

Nótese que, en este caso, los puntos de captura generan una topología lineal, de tal manera que asumimos esta hipótesis sobre las capturas obtenidas, ya que no se quiere tener que lidiar con el problema de tener que representar la topología de una manera útil y tener que extraer caminos lineales a partir de la topología dada.

Se puede encontrar un ejemplo de uso de la herramienta en el anexo B.

3.4 Conclusiones

Por un lado, el diseño y desarrollo del entorno experimental ha permitido implementar topologías de red arbitrarias y generar así un entorno de red con múltiples puntos de captura. Además, esto ha servido para conocer diferentes técnicas de simulación en el ámbito de las redes de comunicaciones, así como parte del funcionamiento de las mismas con tecnologías como OpenFlow. Todo ello se ha implementado mediante una biblioteca llamada **mininetplus** que sobre **mininet** permite simular diferentes topologías definiendo los diferentes elementos a través de su API Python.

Posteriormente, se ha necesitado procesar los datos de varios puntos de captura, es decir, calcular los Δ_i a partir del RTT en varios puntos. A partir de estos datos, se ha procedido a tratarlos convenientemente, viendo si validamos o no la configuración del entorno experimental. En primer lugar, para un rápido análisis y visualización de los datos, se han elaborado prototipos de las herramientas en código Matlab, que tiene una amplia biblioteca por defecto y una alta optimización.

Una vez se ha visto que los datos se correspondían con la configuración del entorno experimental, se ha proseguido con el proyecto implementando una herramienta/biblioteca en Python que admite más opciones, es portable y que está mejor diseñada con menor repetición de código y que es el objetivo principal del trabajo.

No obstante, esto aún deja un tema abierto: ¿Cómo se comporta el entorno experimental? ¿Qué proporciona el análisis? Para responder a estas preguntas y más, el siguiente capítulo estudiará los resultados obtenidos.

4 Resultados y validación

4.1 Introducción

En este capítulo se van a presentar los resultados obtenidos en el proceso de validación. Por brevedad, solo se mostrarán los aspectos más importantes de los resultados para extraer conclusiones. El resto de los resultados se encuentran en el anexo A para más información. Esta tarea es de suma importancia para verificar la funcionalidad de las herramientas y mejoras derivadas de este trabajo.

Se ha seleccionado un conjunto de escenarios de prueba que han permitido evaluar el desarrollo en un amplio abanico de situaciones. En primer lugar, la topología lineal sin congestión, en la que esperamos ver el retardo programado en el entorno experimental de la manera más fiel posible. En segundo lugar, se introducirá congestión adicional para reflejar un entorno más cercano al real. Por último, enfrentaremos el desarrollo a un conjunto de datos procedentes de un despliegue comercial, ilustrando la aplicabilidad del desarrollo realizado a la verificación del estado de red en un entorno real.

4.2 Topología lineal

En esta sección, se analizan los datos provenientes del primer y segundo caso de topologías básicas, estos son, topología lineal con *switches* y con *routers*, ambos sin congestión. En este caso tan sintético, se espera que los resultados se concentren mucho en torno a lo programado en el entorno experimental.

4.2.1 Configuración del entorno y metodología

Las pruebas se han realizado con el entorno experimental desarrollado. Se ha configurado de la siguiente manera:

1. Se crea una topología lineal con 6 nodos (2 nodos terminales y 4 *switches* o *routers*).

2. Se ha configurado con `netem` un retardo en cada enlace de $(10 + 2i)$ms, con i el índice del salto. Todos los enlaces tienen un ancho de banda de 20Mbit/s configurado mediante `tc`.

3. Se abren interfaces de captura en cada uno de los nodos con `tcpdump`. Aunque se podría analizar el tráfico de manera *online*, se evita esto para evitar perturbaciones en el entorno por la carga de la CPU.

4. Se inician varios generadores de tráfico de manera secuencial en el primer nodo que abren conexiones TCP al último nodo. Queremos evitar saturar los enlaces.

5. Se obtienen trazas en los puntos configurados con 5000 conexiones TCP en total para ser procesadas.

4.2.2 Visualización de datos

Una vez hemos obtenido las trazas resultantes del entorno, las pasamos a la herramienta de análisis. En primer lugar, se construyen los superflujos, lo que proporciona muestras del RTT de una misma conexión TCP en varios saltos. Tiene una importancia clave ver la forma de los datos para poder entender los modelos y lo que debemos esperar. Por ello, una parte importante es visualizar tanto los Δ_i como el RTT.

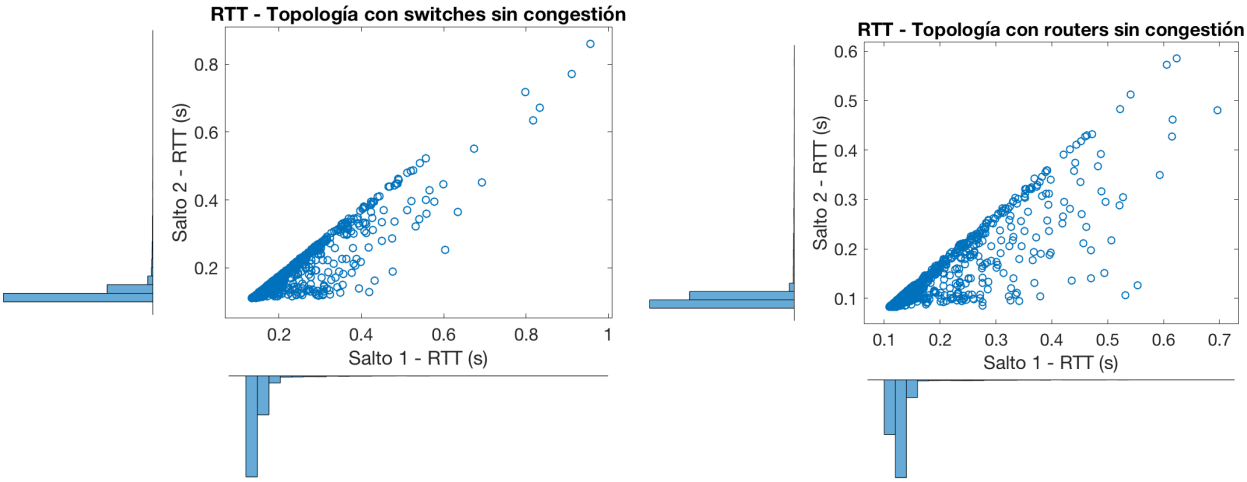

Figura 4.1: Diagramas dispersión de los saltos 1 y 2 para el RTT de la topología lineal con switches (izquierda) y con routers (derecha)

Se observa en la figura 4.1 el RTT para las topologías básicas. Nótese que están limitados todos los diagramas de dispersión por la recta $Y = X$, ya que el $\text{RTT}_i > \text{RTT}_{i+1}$. A priori, se observan distribuciones muy concentradas, como se esperaba, pero con ciertos valores extremos. En cambio, en la figura 4.2 que muestra los Δ_i esta limitación no se ve, pues la forma anterior se ha expandido un poco y se ha reducido la dispersión.

4.2.3 Ajuste de distribuciones

En este caso, para mayor brevedad, solo comentaré los resultados de un salto y del entorno experimental con *switches*. El resto de los resultados son similares al

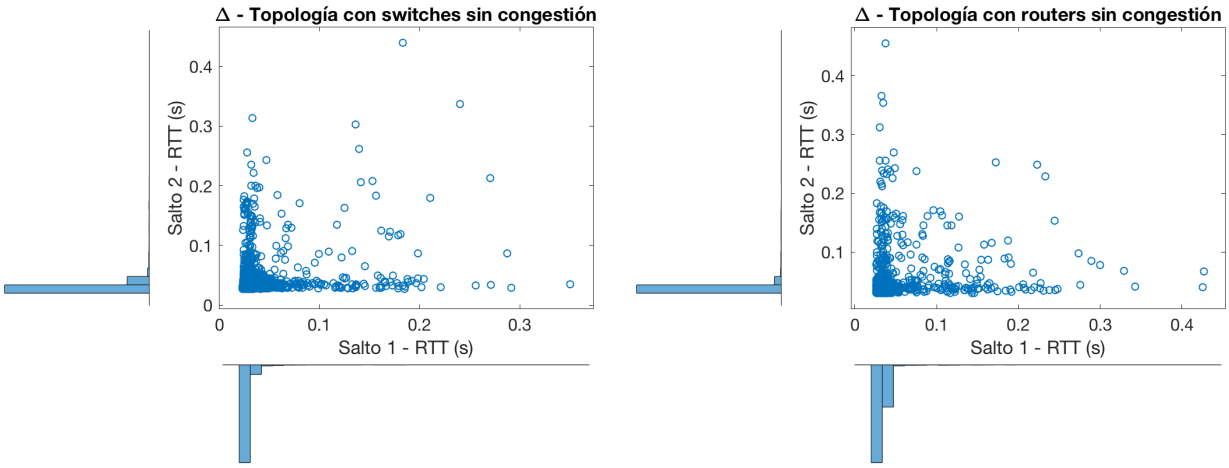

Figura 4.2: Diagramas dispersión de los saltos 1 y 2 para los Δ_i de la topología lineal con *switches* (izquierda) y con *routers* (derecha)

comentado. Estos ajustes han sido obtenidos mediante la herramienta de análisis creada, añadiendo una capa de visualización adicional en la que se muestran los datos y los modelos evaluados con los parámetros estimados.

En la figura 4.3, se observa cómo ajustan las distribuciones a los datos. Para cada uno de los modelos, se proporciona el $\mathbf{R}^2$, el **AIC** y el **BIC** para comparar los modelos entre sí.

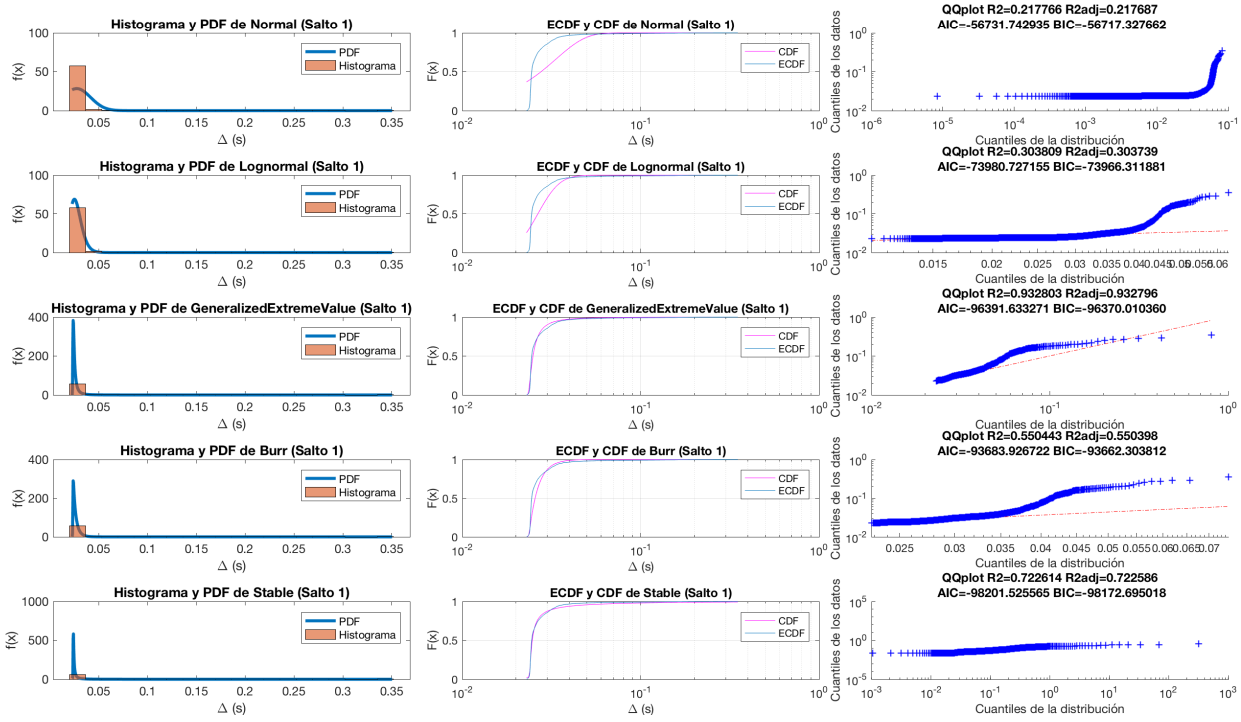

Figura 4.3: PDF, CDF y QQPlot para diferentes modelos de Δ_1 del entorno sin congestión

Se observa como el ajuste solo es razonable para el modelo GEV. Para el resto de casos, el modelo solo es capaz de ajustar una parte de los cuantiles como se muestra en cada uno de los QQPlot.

Esto en principio, son unos resultados un poco desfavorables ya que claramente 4 de los 5 modelos no ajustan en absoluto y un modelo es candidato, aunque observamos una forma ligeramente distinta. Se nota que los modelos no ajustan bien cuando la distribución está muy concentrada en un punto, pero tiene unos pocos valores extremos. Como este no es el caso común de despliegue, es necesario ver los datos con congestión para ver que los modelos sí son razonables cuando hay más tráfico interferente.

4.2.4 Estimación de la moda

Se configuró en el entorno de simulación un retardo en el primer enlace de 12ms en cada sentido. Por lo cual, la moda debería ser ∼24ms. Se muestra en la tabla 4.1. Como los datos no ajustan del todo bien a alguno de los modelos, la moda se desvía del valor programado.

Tabla 4.1: Resultados de la moda para Δ_1 del entorno sin congestión

KDE	**HSM**	**Mod. Normal**
24.153ms	24.190ms	27.615ms
Mod. Lognormal	**Mod. GEV**	**Mod. Burr**
23.811ms	23.990ms	23.999ms

4.3 Topología lineal con congestión

En esta sección vamos a tratar el caso de topología lineal con congestión. En este apartado ya se tiene un objetivo distinto, pues no queremos validar el entorno experimental, sino ver cómo se comportan los equipos de red ante la congestión. En este caso, se espera que los modelos ajusten mejor, ya que las distribuciones no deberían ser tan degeneradas. Como gran parte del proceso es exactamente análogo al caso previo, solo se comentarán los resultados obtenidos y las particularidades de este caso.

4.3.1 Configuración del entorno y metodología

La configuración empleada es igual a la anteriormente utilizado en el caso sin congestión. Simplemente se añade un *host* conectado a cada *switch*. Al configurar el mismo entorno, también queremos ver cómo cambia el añadir congestión al entorno, esto es, cómo cambia la moda y cómo varía el ajuste de los modelos.

Se han probado diferentes maneras de generar congestión, como tráfico HTTP, utilización de `ping` ICMP, la ejecución de herramientas de medición de ancho de banda como `iperf` o sistemas más sofisticados para generación de tráfico con patrones reales como FlexiTOP. Ya que saturar el enlace solo generaba perdidas, se descarta el uso de `iperf` y se han configurado el resto de métodos para no llegar a la saturación total del enlace. Por ello, una manera razonable es introducir cierta aleatoriedad a esta generación de tráfico.

Por simplicidad, y ya que los resultados con los diferentes métodos eran similares, estos nuevos nodos se programan para que envíen `pings` de tamaño

1000Bytes en intervalos aleatorios, pero de un orden muy pequeño (10-100ms). Además, en los puntos de captura se utiliza un filtro de captura para evitar que el hecho de capturar una cantidad mayor de tráfico nos afecte a la parte de captura de las trazas.

4.3.2 Visualización de datos

En la figura 4.4 se muestran los diagramas de dispersión del RTT y de los Δ_i. Existe una relación sublineal en la distribución de los RTT$_i$. Si estos resultados se comparan con los mismos en ausencia de congestión, la dispersión ha aumentado y las distribuciones han variado significativamente. En este caso, las distribuciones muestran una nube de puntos muy concentrada en torno a la moda, con ciertos *outliers*. En este caso, previo al estudio de distribuciones, parece que van a tener un carácter menos degenerado que las anteriores.

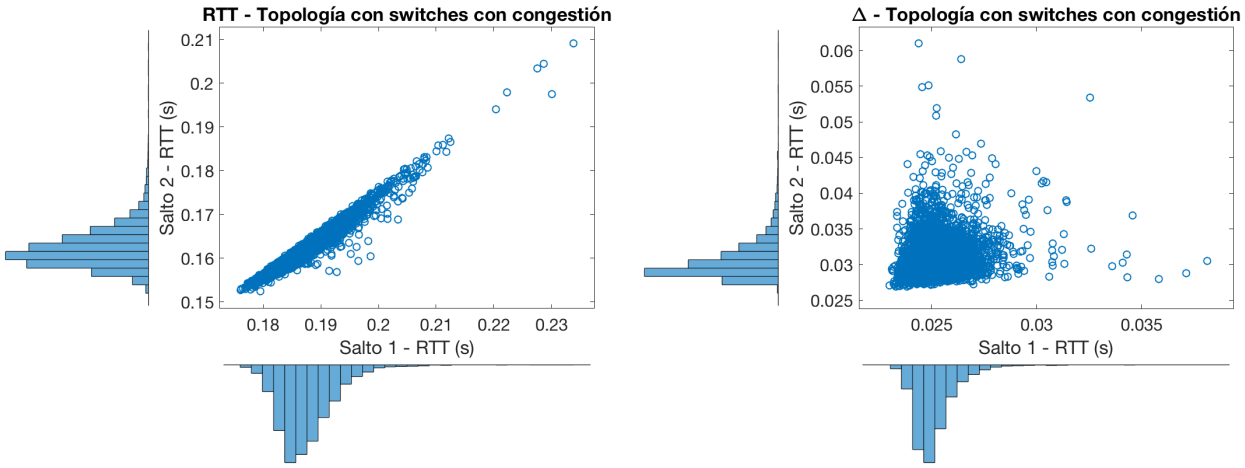

Figura 4.4: Diagramas dispersión de los saltos 1 y 2 para el RTT (izquierda) y los Δ_i (derecha) de la topología lineal con congestión

4.3.3 Ajuste de distribuciones

Para este caso, el ajuste a los modelos es bien diferente del escenario sin congestión. En la figura 4.5, se ve como los modelos normal y lognormal no se ajustan del todo, pero el modelo GEV, el de Burr y el α-estable (con $\mathbf{R}^2 > 0.95$) se aproximan lo suficiente como para considerar que, salvo por los percentiles más en la cola, se ajustan a la distribución modelada.

En este caso, los modelos han probado su utilidad y se observa que según el $\mathbf{R}^2$ el modelo elegido es el α-estable, el más complejo; según el **AIC**, el modelo elegido es el GEV y según el **BIC**, el modelo elegido es el de Burr. Se recuerda que el $\mathbf{R}^2$ solo tiene en cuenta el ajuste, el **AIC** el ajuste y el número de parámetros y el **BIC** el ajuste y el número de parámetros ajustado con el tamaño de la muestra.

Esto nos permite elegir el modelo que más ajusta teniendo en cuenta distintos factores que pueden afectar a la aplicación de monitorización. Si se desea un modelo en tiempo real, tanto el modelo de Burr como el modelo GEV resultan adecuados. Si deseamos tener el modelo que mejor ajuste, podemos utilizar la distribución α-estable, que nos va a proporcionar un modelo cuyo cómputo es complejo, pero que obtiene el mejor ajuste.

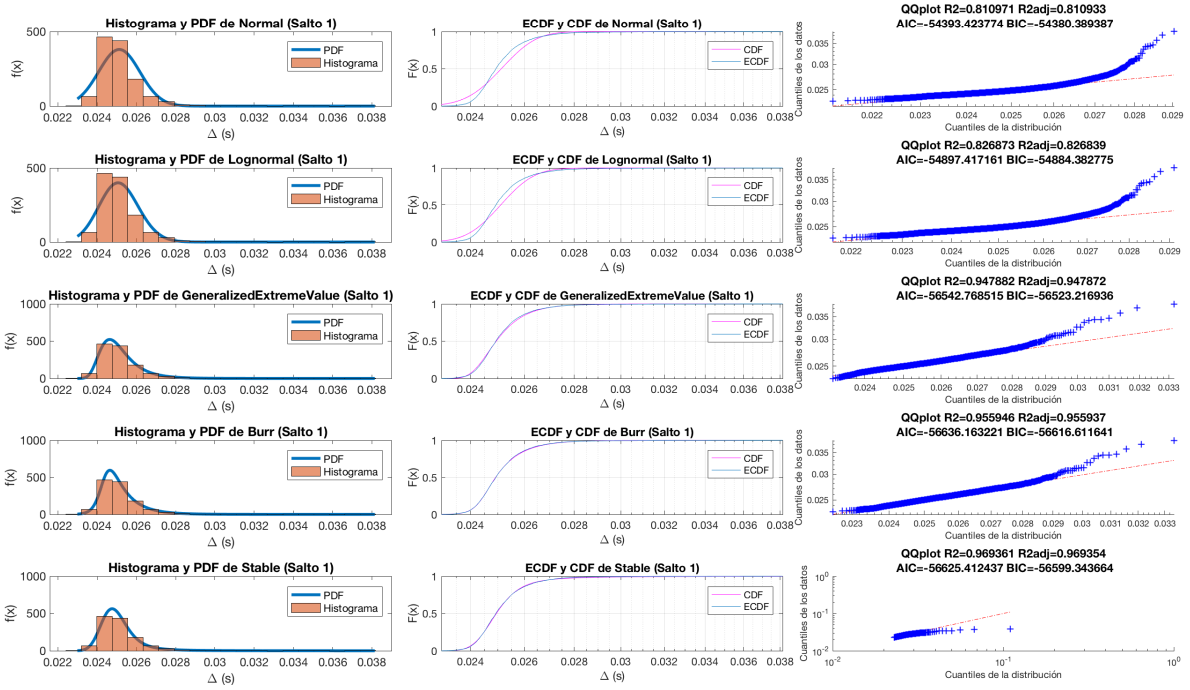

Figura 4.5: PDF, CDF y QQPlot para diferentes modelos de Δ_1 del entorno con congestión

4.3.4 Estimación de la moda

En la tabla 4.2, se puede ver como la moda en todos los modelos y también estimada con HSM y como el máximo es 24.7ms. Es decir, los 24ms programados más otra componente adicional de la congestión de 0.7ms.

Tabla 4.2: Resultados de la moda para Δ_1 del entorno con congestión

KDE	HSM	Mod. Normal
24.669ms	24.699ms	25.125ms
Mod. Lognormal	**Mod. GEV**	**Mod. Burr**
24.665ms	24.640ms	24.650ms

Se observa una relación entre el ajuste de la distribución a los datos y la moda obtenida si usamos el modelo. En este caso, la media (moda del modelo normal) no sufre una gran desviación debido al número de muestras y a la poca presencia de muestras con valores extremos.

4.4 Datos reales (*Dataset* 1)

4.4.1 Descripción del escenario

Este conjunto de datos, que se denominará ***dataset* 1**, ha sido proporcionado por naudit HPCN, empresa de base tecnológica de las universidades Autónoma de Madrid y Pública de Navarra. El conjunto de datos tiene las siguientes características:

- Extraído de una operadora de red.
- Contienen tráfico de *switches* troncales, *switches* de servicio, balanceadores de carga y granjas de máquina virtuales.

- Provienen de infraestructura física y virtualizada.
- Ninguna de estas trazas ha sido capturada en un equipo dedicado exclusivamente a la monitorización de red, sino del propio equipamiento de red con el software de captura que traen integrado.

Debido a la presencia de unos pocos *outliers* en el segundo salto, se han preprocesado los datos para que las visualizaciones sean correctas y representativas del comportamiento de estos equipos, aunque los flujos que han resultado *outliers* son probablemente objeto de futuros análisis. Debido a que los destinos de los flujos son máquinas virtuales, se tiene la hipótesis de que fueran los flujos iniciales que arrancasen las máquinas virtuales o peticiones de una naturaleza muy distinta a la normal de los servicios.

4.4.2 Visualización de datos

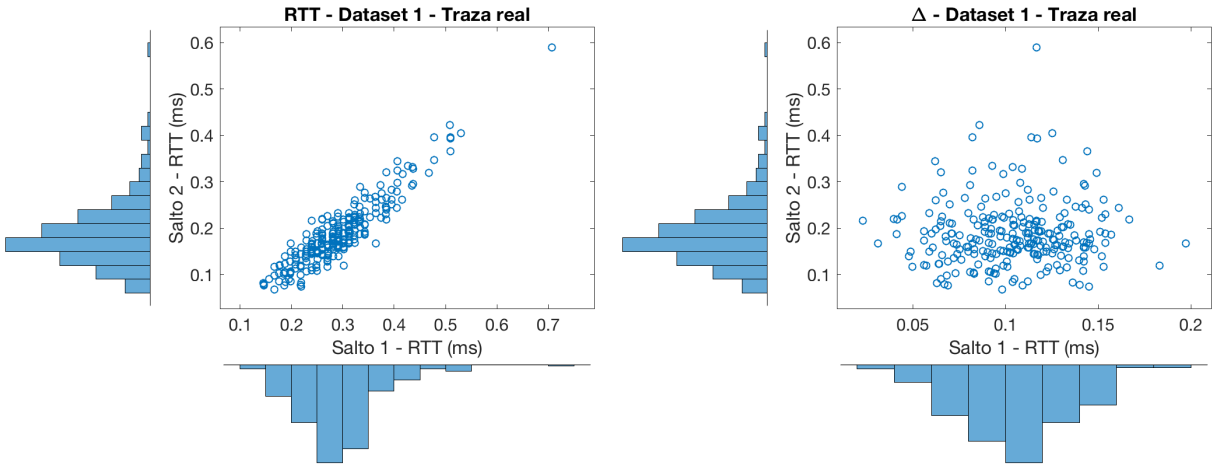

Figura 4.6: Diagramas dispersión de los saltos 1 y 2 para el RTT (izquierda) y los Δ_i (derecha) del *dataset* 1 tras la eliminación de *outliers*.

Se muestra la distribución de los datos en la figura 4.6. En primer lugar, se observa como los órdenes de magnitud son distintos, ya que esta infraestructura está muy próxima físicamente. En segundo lugar, se nota que la eliminación de los *outliers* en Δ_1 no era necesaria, ya que se observa a priori una forma casi normal. No obstante, en Δ_2, se observa aún restos de la cola existente de la distribución y que hemos eliminado para ver correctamente la distribución de la parte más concentrada. En los siguientes apartados, no se eliminarán estos *outliers* completamente para ver si las distribuciones utilizadas modelan adecuadamente la cola de la distribución de Δ_2.

Se nota, comparando RTT con Δ_i, que la distribución conjunta cambia mucho y en este caso, la conjetura de independencia parece razonable ya que la dependencia entre los Δ_i no se observa en la nube de puntos.

4.4.3 Independencia

Queremos comprobar la hipótesis nula: $H_0 : \Delta_1$ y Δ_2 son independientes. Para ello, calculamos el estadístico del contraste como se ha explicado en el capítulo 2.

Por otro lado, no solo necesitamos el estadístico sino también un criterio de aceptación o rechazo. Para ello, se utiliza *Bootstrapping* para estimar el valor crítico a nivel de significación $\alpha = 0.05$ tal que rechazamos H_0.

Tabla 4.3: Resultados de independencia para Δ_1 y Δ_2 en el *dataset* 1

Estadístico HSIC	Valor crítico $\alpha = 0.05$
$9.4821418 \cdot 10^{-18}$	$4.7421118 \cdot 10^{-16}$

Ya que el estadístico es menor que el umbral de rechazo (ver tabla 4.3), **no** tenemos suficiencia estadística para rechazar H_0. Esto es bastante positivo para el estado de la red, ya que podemos asumir la independencia del retardo.

4.4.4 Modelado mediante distribuciones

Sabiendo que las componentes Δ_i son independientes, podemos estudiarlas por separado como dos componentes que constituyen el RTT observado. Por ello, se van a estudiar diferentes modelos para cada componente.

Para ver el modelo que mejor ajusta a la distribución de Δ_1, se muestran en la figura 4.7 los diferentes modelos ajustados para esta variable comparados con los datos muestrales.

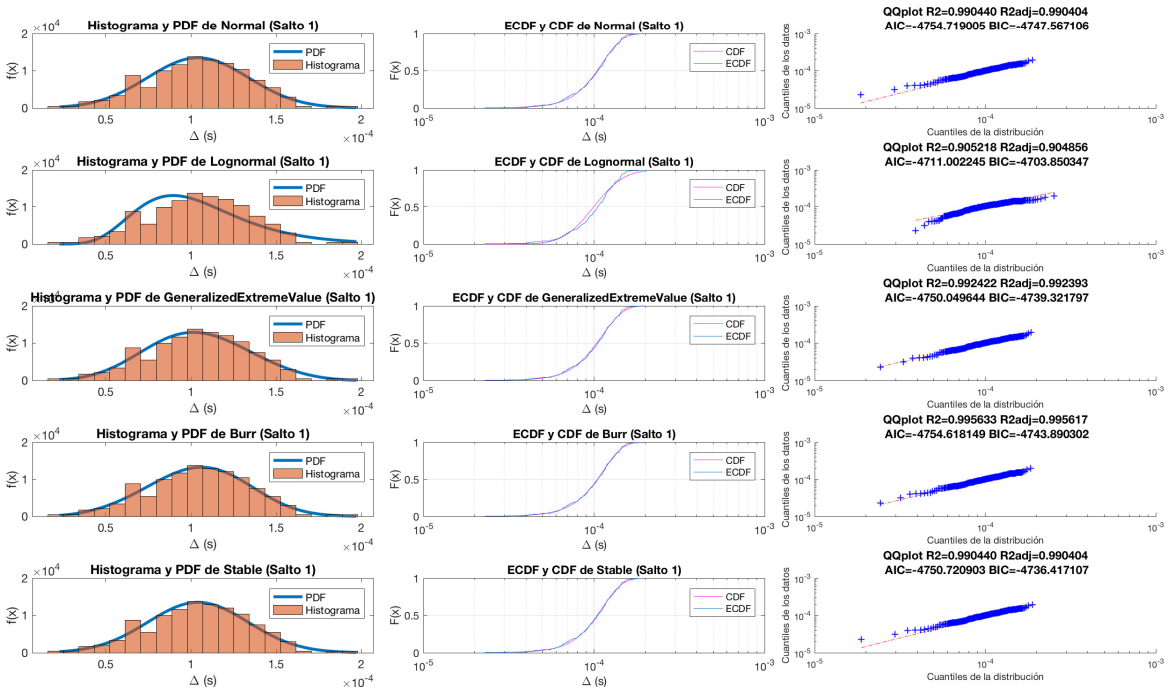

Figura 4.7: PDF, CDF y QQPlot para diferentes modelos de Δ_1 del *dataset* 1

En esta misma figura 4.7, se observa como el modelo que mejor ajusta es el modelo de Burr, pues el $\mathbf{R}^2$ es el más alto. No obstante, **AIC** y **BIC** compensan la complejidad del modelo y con estas correcciones el modelo elegido es el modelo Normal. Esto nos viene a indicar la buena salud de este segmento de red, que carece de valores extremos.

Con esto hemos deducido que Δ_1, la componente de red entre los dos puntos de captura, influye de manera normal en el RTT. No obstante, vemos como los

otros modelos para estos casos en los que no encontramos eventos extremos, resultan también razonables.

Por otro lado, en la figura 4.8, se muestra un caso totalmente distinto en el que la distribución no se asemeja nada a una normal, siendo el único modelo que ajusta razonablemente el modelo α-estable.

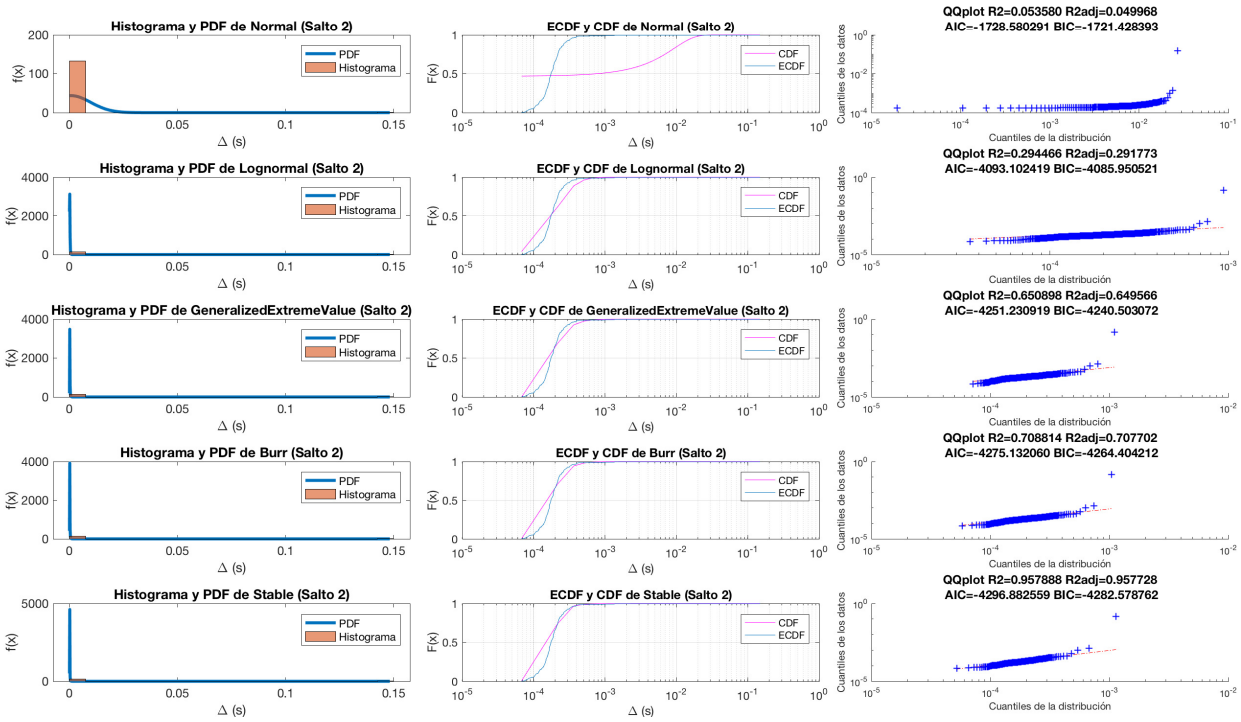

Figura 4.8: PDF, CDF y QQPlot para diferentes modelos de Δ_2 del *dataset* 1

4.4.5 Estimación de la moda

Para la estimación de la moda, usamos todos los métodos descritos anteriormente y los comparamos. Se muestran los resultados en la tabla 4.4. Se observa un fenómeno curioso: las distribuciones que no ajustan bien la cola, tienen variaciones muy grandes de la moda. En todos los casos, la distribución de Burr, el algoritmo HSM y el estimador directo a través del KDE resultan en cifras muy similares.

Tabla 4.4: Resultados de la moda para Δ_1 y Δ_1 del *dataset* 1

Δ_1			Δ_2		
KDE	**HSM**	**Mod. Normal**	**KDE**	**HSM**	**Mod. Normal**
0.1080ms	0.1124ms	0.1042ms	0.1618ms	0.1669ms	0.7553ms
Mod. LogN	**Mod. GEV**	**Mod. Burr**	**Mod. LogN**	**Mod. GEV**	**Mod. Burr**
0.0861ms	0.1055ms	0.1063ms	0.1223ms	0.1251ms	0.1546ms

Además, esto indica que estimar mediante la media (que es la moda en el caso del ajuste a un modelo normal y por tanto se muestra en la tabla como ese modelo) resulta bastante inexacto si tenemos una cola pesada o la presencia de *outliers*. Este último caso es observable en el caso de Δ_2, en el que el valor obtenido de la media está incluso muy lejos del lugar en el que se concentra la distribución.

4.5 Conclusiones

En este capítulo se ha probado como el *testbed* implementado nos permite simular tráfico con capturas en varios puntos, lo que posibilita evaluar las herramientas en escenarios controlados. Además, las herramientas elaboradas con las ideas de todo este trabajo proveen de una información que hasta ahora era muy difícil de obtener.

A la luz de los resultados sobre datos reales, hemos podido confirmar hipótesis de buen funcionamiento (independencia, ausencia de colas pesadas, normalidad) sobre el equipamiento de red que nos interesaba mediante un análisis pasivo del tráfico de red. Gracias a estos resultados, se pueden elaborar modelos predictivos que permitan determinar cuando este segmento de red se encuentre más saturado de lo normal.

Estos modelos nos aportan además información de cómo es la naturaleza de funcionamiento del equipamiento de red (asimetrías, colas pesadas, etc.) y, por su simplicidad, nos permite implementarlos en tiempo real para sistemas de gestión de red, detección de anomalías o aprovisionamiento automático de recursos.

En resumen, estos resultados confirman como la herramienta de análisis es muy útil para la monitorización pasiva de redes, descomponiendo el RTT en componentes más simples que contienen el retardo que introduce cada segmento de red concreto.

5 Conclusiones

5.1 Conclusiones

En este trabajo de fin de grado se ha realizado un estudio de las posibilidades del análisis de datos de red en múltiples saltos. Se parte de la idea de que $\text{RTT}_k = \sum_{i=k}^{N} \Delta_i$ y por tanto $\Delta_i = \text{RTT}_i - \text{RTT}_{i+1}$. Así, se ha examinado el estado del arte, en el que se ha investigado y profundizado en el área de la monitorización de red, se han extraído conceptos clave que se utilizan o generalizan para el caso que se ha estudiado. En este caso, nos hemos centrado en independencia de las componentes como condición de salud de la red, modelos para la caracterización de estas componentes y la moda como valor más frecuente y representativo de los datos.

Para la reproducción de experimentos de red con entorno multi-captura y el análisis de datos de múltiples saltos, se han elaborado dos herramientas distintas en las que se proporciona toda la funcionalidad requerida para desarrollar este trabajo. En el primer caso, se ha creado una biblioteca sobre **mininet**, que ha permitido reproducir mediante código Python los experimentos de red con un nivel de reconfiguración muy alto. Para la componente de análisis de datos ha dado lugar a una herramienta de análisis de datos de múltiples saltos.

En resumen, y a la luz de los resultados obtenidos, se han hallado nuevas maneras de observar el RTT en una red con la simple transformación en las componentes Δ_i, con modelos matemáticos sencillos que permiten obtener información que ayuda a la planificación de la red, detección de anomalías y aprovisionamiento de los recursos. Esto permite obtener indicadores de saturación en nuestra infraestructura de red, permitiendo así no solo diagnosticar si hay un problema sino además proporcionar la localización o localizaciones del problema. Esto abre una nueva ventana a la monitorización de los proveedores de servicio de Internet o de servicios *cloud*, ya que con estas herramientas es posible averiguar si son ellos los responsables de la mala QoS, o bien lo es el cliente o incluso el proveedor de servicio que ha alquilado los servidores.

Además, de manera pasiva y utilizando protocolos abiertos y muy extendidos como NetFlow o IPFIX, se puede planificar y escalar los recursos en base a modelos que no tienen una base heurística sino una fuerte base matemática en la que se cuantifica la probabilidad de que un elemento se desvíe una cierta cantidad de la media o se emplean valores más robustos frente a grandes desviaciones.

5.2 Contribuciones

Esta herramienta de análisis y su tecnología han sido trasferidas a la industria. En concreto, esta actividad de transferencia se encuadra dentro del marco de colaboración de la Cátedra UAM-naudit, estando en la actualidad en explotación por parte de naudit para el análisis de datos de red a nivel comercial. Además, dentro del entorno de pruebas, se ha utilizado FLEXITOP, un sistema que desarrollé para la generación de tráfico y monitorización activa de servicios Over-The-Top que actualmente se encuentra desplegada en distintos equipos comerciales y ha dado lugar a dos publicaciones [30, 31].

Además, se ha creado una librería, **mininetplus**, que se va a reaprovechar para la elaboración de entornos experimentales en futuras ocasiones y que se encuentra publicada en un repositorio de github asociado al grupo de investigación HPCN-UAM.

5.3 Trabajo futuro

Este trabajo contiene un amplio estudio de diferentes técnicas y modelos. Sin embargo, hay temas interesantes que se estudiarán más adelante y que, por tanto, conformarán el trabajo futuro.

Un tema que ha sido central para este trabajo ha sido el concepto de moda y su uso como valor robusto para modelar ciertas distribuciones. No obstante, el concepto de moda no es algo que se presenta de manera única en los datos reales, por lo que en general se podría extender el problema anterior al de hallar las modas. En este caso, los modelos evolucionan hacia modelos con mixturas [14] o la separación de las diferentes modas para su estudio por separado. Pese a la apariencia inocente, hallar las modas de una distribución resulta un proceso más complejo, en el que el primer paso, que sería contar las modas, no resulta simple. Técnicas como HSM no solucionan este problema, por lo que habrá que centrarse en maneras de ver ese problema desde el estimador KDE y posteriormente hallar una optimización mediante ideas parecidas a las de HSM o totalmente renovadas.

Por otro lado, en la utilización de modelos, hemos utilizado un catálogo de unos pocos modelos viendo el mejor ajuste. En todo momento, se ha trabajado con condiciones necesarias para ver el ajuste, de cara a elaborar un *ranking* o hallar el mejor de los ajustes. Sin embargo, en ningún momento hemos comprobado la suficiencia, esto es, que el mejor modelo ajusta a un nivel de confianza α. Esto es otro problema totalmente distinto, pero que nos puede ayudar a entender la naturaleza intrínseca al tráfico de red.

Por último, en los casos reales y de monitorización continua, interesa no solo ver el modelo en diferentes intervalos concretos de tiempo sino además ver como varían los parámetros del modelo respecto al tiempo [4]. Esto posibilita el ver como en el tiempo, los modelos se adaptan a los eventos que vayan sucediendo, agregando entonces la dimensión temporal al modelo que resulta ideal para ciertas tareas como la detección de anomalías.

BIBLIOGRAFÍA

[1] G. Almes, S. Kalidindi, and M. J. Zekauskas. A Round-trip Delay Metric for IPPM. RFC 2681, Sept. 1999.

[2] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 153–158, New York, NY, USA, 2006. ACM.

[3] D. R. Bickel and R. Frühwirth. On a fast, robust estimator of the mode: Comparisons to other robust estimators with applications. *Computational Statistics & Data Analysis*, 50(12):3500 – 3530, 2006.

[4] C. Bizumuremyi. Análisis de Tráfico en Internet Utilizando Distribuciones Alfa estables en procesadores paralelos, 2017.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[6] B. Claise, B. Trammell, and P. Aitken. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information RFC 7011, 2013.

[7] T. L. Foundation. The Linux Foundation - Open Source Networking. https://www.linuxfoundation.org/projects/networking/. Última vez accedido: 21-05-2018.

[8] V. S. Frost and B. Melamed. Traffic modeling for telecommunications networks. *IEEE Communications Magazine*, 32(3):70–81, March 1994.

[9] S. Garcia-Jimenez, E. Magaña, D. Morató, and M. Izal. Pamplona-traceroute: Topology discovery and alias resolution to build router level Internet maps. In *Global Information Infrastructure Symposium - GIIS 2013*, pages 1–8, Oct 2013.

[10] A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola. A Kernel Statistical Test of Independence. In *Proceedings of the 20th International*

Conference on Neural Information Processing Systems, NIPS'07, pages 585–592, USA, 2007. Curran Associates Inc.

[11] A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola. A Kernel Statistical Test of Independence. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 585–592. Curran Associates, Inc., 2008.

[12] H. Jiang and C. Dovrolis. Passive Estimation of TCP Round-trip Times. *SIGCOMM Comput. Commun. Rev.*, 32(3):75–88, July 2002.

[13] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [*Online*; Accedido por última vez el 09/06/2018].

[14] G. Julián. Automatic estimation of mixtures of α-stable distributions. Master's thesis, École Polytechnique Fédérale de Lausanne, 2017.

[15] S. Kalidindi, M. J. Zekauskas, and D. G. T. Almes. A One-way Delay Metric for IPPM. RFC 2679, Sept. 1999.

[16] B. Lantz and B. O'Connor. A Mininet-based Virtual Testbed for Distributed SDN Development. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 365–366, New York, NY, USA, 2015. ACM.

[17] J. Liebeherr, A. Burchard, and F. Ciucu. Delay Bounds in Communication Networks With Heavy-Tailed and Self-Similar Traffic. *IEEE Transactions on Information Theory*, 58(2):1010–1024, Feb 2012.

[18] M. Martínez. Desarrollo de una herramienta de tratamiento estadístico de distribuciones alfa-estables para su aplicación al análisis de tráfico de redes IP. Master's thesis, Universidad Autónoma de Madrid, 2017.

[19] D. McFadden. Modelling the Choice of Residential Location. Cowles Foundation Discussion Papers 477, Cowles Foundation for Research in Economics, Yale University, 1977.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[21] R. V. D. Meent, M. Mandjes, and A. Pras. Gaussian traffic everywhere? In *2006 IEEE International Conference on Communications*, volume 2, pages 573–578, June 2006.

[22] F. Morales, M. Ruiz, L. Gifre, L. M. Contreras, V. Lopez, and L. Velasco. Virtual network topology adaptability based on data analytics for traffic prediction. *IEEE/OSA Journal of Optical Communications and Networking*, 9(1):A35–A45, Jan 2017.

[23] D. Muelas, J. E. López de Vergara, and J. R. Berrendero. Functional Data Analysis: A step forward in Network Management. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 882–885, May 2015.

[24] D. Muelas, J. E. López de Vergara, J. R. Berrendero, J. Ramos, and J. Aracil. Facing Network Management Challenges with Functional Data Analysis: Techniques & Opportunities. *Mobile Networks and Applications*, 22(6):1124–1136, Dec 2017.

[25] D. Muelas, J. Ramos, and J. E. López de Vergara. Software-Driven Definition of Virtual Testbeds to Validate Emergent Network Technologies. *Information*, 9(2), 2018.

[26] T. E. Oliphant. *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015.

[27] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, Jun 1995.

[28] K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[29] D. Pediaditakis, C. Rotsos, and A. W. Moore. Faithful Reproduction of Network Experiments. In *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '14, pages 41–52, New York, NY, USA, 2014. ACM.

[30] D. Perdices, J. E. López de Vergara, P. Roquero, C. Vega, and J. Aracil. FlexiTop: sistema de medidas de calidad de servicio escalable y flexible para servicios OTT. In *XIII Jornadas de Ingenieria Telematica - JITEL2017*, pages 1–6, 09 2017.

[31] D. Perdices, J. E. López de Vergara, P. Roquero, C. Vega, and J. Aracil. FlexiTop: a flexible and scalable network monitoring system for Over-The-Top services. *Network Protocols and Algorithms*, 9:136, 02 2018.

[32] I. Prieto, M. Izal, E. Magaña, and D. Morató. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, INTERNET 2015, 2006.

[33] E. Rojas, R. Doriguzzi-Corin, S. Tamurejo, A. Beato, A. Schwabe, K. Phemius, and C. Guerrero. Are We Ready to Drive Software-Defined Networks? A Comprehensive Survey on Management Tools and Techniques. *ACM Comput. Surv.*, 51(2):27:1–27:35, Feb. 2018.

[34] S. Seabold and J. Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[35] C. Vega. *Explorando el proceso de recolección, análisis y visualización del tráfico en las redes de computadoras*. PhD thesis, Universidad Autónoma de Madrid, 2018.

RESULTADOS EXTENDIDOS

A

Se muestra a continuación una colección de los resultados del resto de saltos de los diferentes conjuntos de datos.

A.1 Topología linear de conmutadores sin congestión

Como ya se han mostrado las distribuciones del primer salto, se incluye a continuación las figuras de cada uno de los saltos:

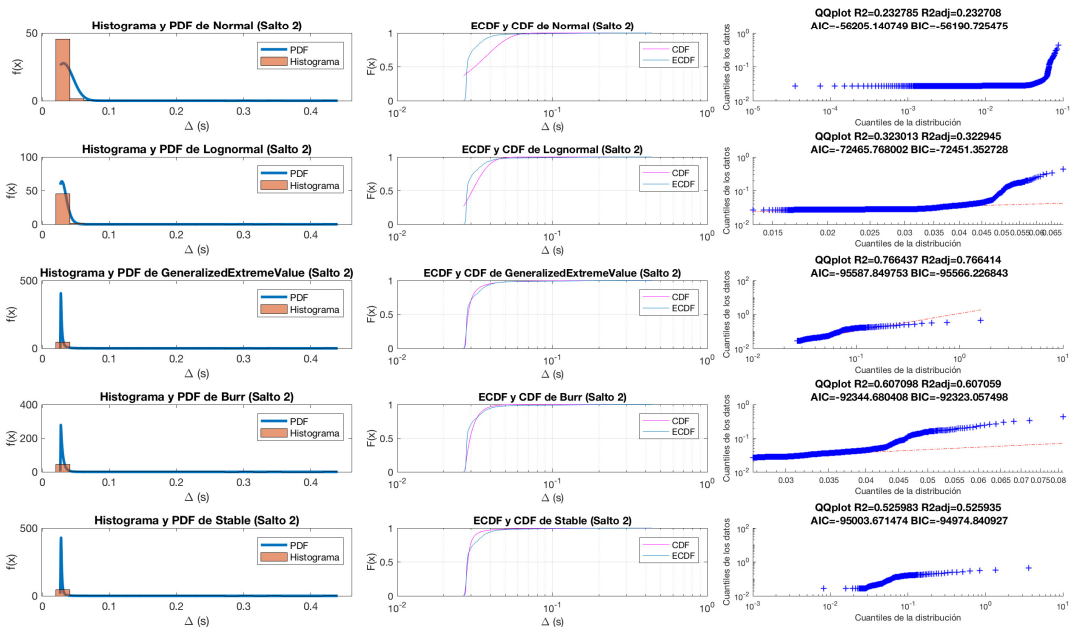

Figura A.1: PDF, CDF y QQPlot para diferentes modelos de Δ_2 del entorno sin congestión y con conmutadores

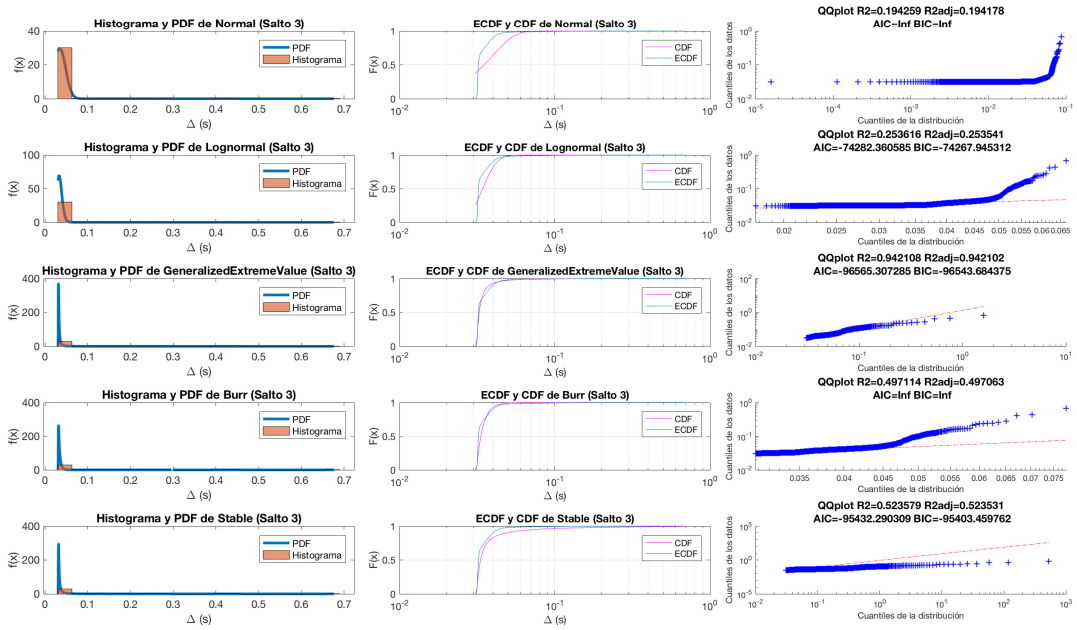

Figura A.2: PDF, CDF y QQPlot para diferentes modelos de Δ_3 del entorno sin congestión y con conmutadores

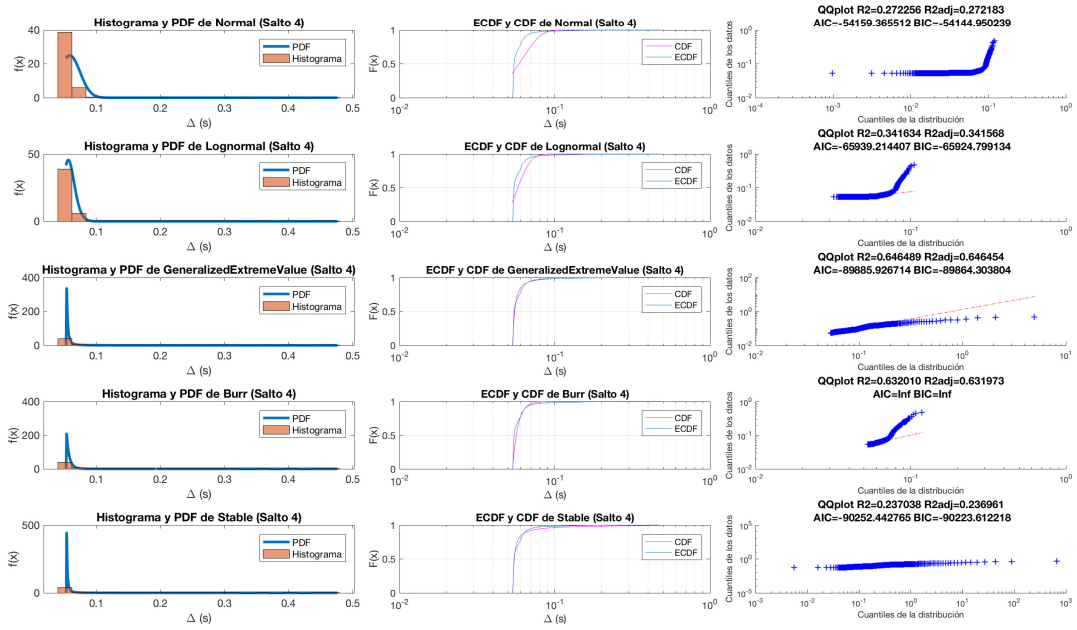

Figura A.3: PDF, CDF y QQPlot para diferentes modelos de Δ_4 del entorno sin congestión y con conmutadores

Los datos obtenidos son similares al primer salto. En ocasiones el AIC o BIC son infinito debido a que el logaritmo de un valor muy próximo a cero es $-\infty$ a efectos del motor de cálculo numérico lo que sucede cuando se calcula la log-verosimilitud.

A.2 Topología linear con congestión

De nuevo, se omite el primer salto que era el caso de estudio de la sección correspondiente del documento

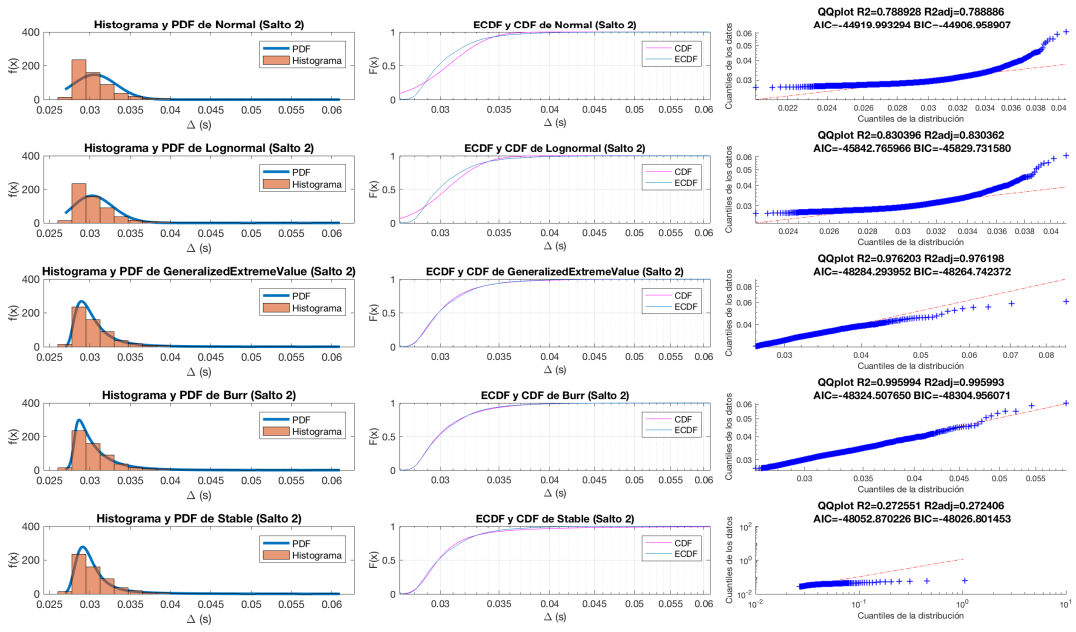

Figura A.4: PDF, CDF y QQPlot para diferentes modelos de Δ_2 del entorno con congestión

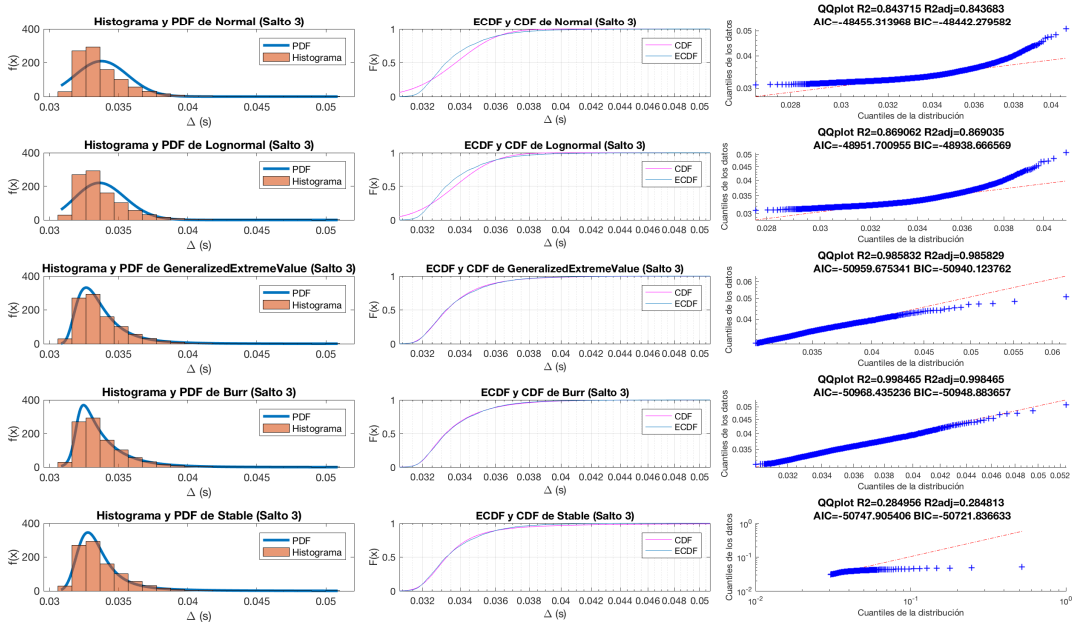

Figura A.5: PDF, CDF y QQPlot para diferentes modelos de Δ_3 del entorno con congestión

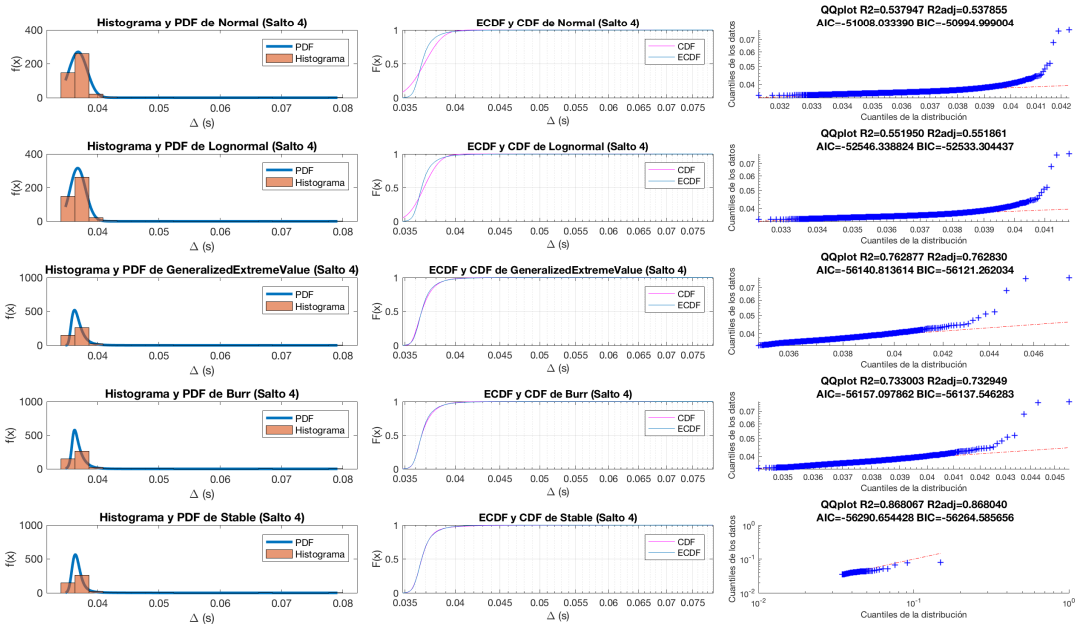

Figura A.6: PDF, CDF y QQPlot para diferentes modelos de Δ_4 del entorno con congestión

Destaca que, en este último salto (ver figura A.6), los modelos no ajustan de manera tan excepcional como en los anteriores, ya que en este caso el tiempo de procesado del servidor afecta a la distribución observada.

B

EJEMPLO DE USO Y SALIDA DE LA HERRAMIENTA DE ANÁLISIS

En este anexo, se incluye un ejemplo de uso de la herramienta en su versión estable 1.1.

```
$ multiFlowCorrelator -h
usage: multiFlowCorrelator [-h] -o OUTPUT N [N ...]

Create a superflow file and process all data

positional arguments:
  N                     Files to be processed. Either pcap
(pattern *.pcap) or flows (pattern *.csv or *.txt).

optional arguments:
  -h, --help            show this help message and exit
  -o output, --output output

$ multiFlow -o superflujos.csv traza1.pcap traza2.pcap

Column 0
logN: s=1.658271, loc=0.018027, scale=0.000348
logN: mode=1.804920e-02, median=0.018375, mean=0.019405, std=0.005272,
AIC=-60369.682951

KDE:  mode=1.816562e-02, median=0.018040, mean=0.019295, std=0.003539
HSM:  mode=2.548945e-02 (exponential)
HSM:  mode=1.842499e-02 (arithmetical)
GEV:  c=-1.376686, loc=0.018190, scale=0.000234
GEV:  mode=1.807178e-02, median=0.018301, AIC=-60275.514513
Stbl: alpha=0.367269, beta=1.000000, loc=0.018025, scale=0.000651
Stbl: mode=nan, median=0.019536, mean=0.018025, std=0.000920,
AIC=-57293.927244
```

```
Column 1
logN: s=0.619467, loc=0.133310, scale=0.013036
logN: mode=1.421913e-01, median=0.146346, mean=0.149103, std=0.010802,
AIC=-33995.404309
KDE:  mode=1.412705e-01, median=0.135815, mean=0.149313, std=0.014183
HSM:  mode=1.429345e-01 (exponential)
HSM:  mode=1.421810e-01 (arithmetical)
GEV:  c=-0.252256, loc=0.143874, scale=0.005943
GEV:  mode=1.425748e-01, median=0.146156, AIC=-34035.428807
Stbl: alpha=0.960289, beta=0.995335, loc=0.074532, scale=0.004335
Stbl: mode=nan, median=0.146223, mean=0.074532, std=0.006131,
AIC=-33434.539342

$ tail -n1 superflujos.csv # Este fichero son los Delta_i
2.263188362099999990e-02 1.434669494630000097e-01
```